COMMODORE 64

i file

Rita Bonelli

GRUPPO
EDITORIALE
JACKSON

COMMODORE 64
i file

di
Rita Bonelli

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

INDICE

PREFAZIONE

Il presente volume è dedicato ai file di dati su dischetto e su cassetta. Esso fa seguito al primo volume, dedicato al BASIC, e a quello faccio spesso riferimento, senza ripetere cose già dette.

Il libro si articola in tre capitoli: il primo di carattere generale, il secondo dedicato alla cassetta e il terzo al dischetto. Sono presenti numerosi esempi, realizzati con la tecnica dei sottoprogrammi, per consentire al lettore di operare facilmente le modifiche necessarie per adattarli alla risoluzione dei suoi problemi.

I file sono trattati servendosi esclusivamente del linguaggio BASIC, e, mi sembra interessante mostrare, come si possano capire molte cose sul funzionamento delle periferiche, anche usando solo un linguaggio evoluto.

Nelle tre appendici faccio un breve cenno ad alcuni programmi in commercio; si tratta di pacchetti formati da programmi scritti in linguaggio ASSEMBLER. Essi risultano indubbiamente più veloci ed efficienti dei programmi scritti in BASIC.

Mi sembra importante sottolineare che, anche coloro che si dedicano con interesse allo studio della programmazione, è bene si abituino ad usare il software disponibile sul mercato. Si trovano ottimi programmi, sia tra gli applicativi, che tra quelli di aiuto nella programmazione come TOOL 64 e THE MANAGER, distribuiti dalla COMMODORE. Non posso non accennare agli utilissimi programmi di elaborazione dei testi (WORD PROCESSING), e in particolare a EASY SCRIPT, diponibile per il COMMODORE 64.

La filosofia che ho applicato nella stesura di questo libro è sempre la stessa. Dopo aver capito i principi fondamentali che stanno alla base dell'informatica, bisogna convalidare i concetti lavorando su un calcolatore.

Per lavorare su un calcolatore ci vuole pazienza e curiosità, ma vale la pena di provare, perchè si ricavano molte soddisfazioni. Inoltre, è bene ricordare che, tramite la "macchina", si è sempre in contatto con altri "umani", coloro che ne hanno progettato l'hardware e il software e sono riusciti a realizzarla.

(Rita Bonelli)

L'autrice ringrazia James Bachmann, Amministratore Delegato, e Sergio Messa, Direttore Generale della Commodore Italiana s.p.a., per avere messo a disposizione le apparecchiature e la documentazione necessarie alla realizzazione dell'opera.

Tutti i programmi che compaiono nel testo sono stati provati sul calcolatore e sono stati memorizzati su un dischetto. L'editore mette in vendita questo libro in due confezioni:

- libro + dischetto,
- solo libro.

Il dischetto dei programmi del libro viene fornito dall'editore anche a richiesta. Dato che i programmi relativi alla cassetta sono pochi, non è stata preparata una cassetta registrata. Tali programmi sono registrati sul dischetto.

I numeri esadecimali che compaiono nel testo, salvo avviso contrario, sono contraddistinti da una H finale.

CAPITOLO 1

I FILE

1.1 Cosa è un file

Un file è un insieme di registrazioni, che, per un motivo qualunque, si desidera raggruppare.
Le registrazioni possono essere effettuate su un qualunque supporto fisico; i più comuni supporti fisici sono:

1) fogli di carta,
2) nastri di carta,
3) nastri magnetici,
4) dischi magnetici,
5) tamburi magnetici.

Sul supporto di tipo 1 le registrazioni avvengono sotto forma di caratteri direttamente leggibili, o disegni ottenuti per punti.
Sul supporto di tipo 2 le registrazioni avvengono sotto forma di fori, che rappresentano i caratteri in un codice. Se si è molto pazienti, tali registrazioni possono anche essere lette direttamente.
Sui supporti di tipo 3, 4 e 5 le registrazioni sono magnetiche e quindi non leggibili direttamente.

In questo libro ci occupiamo dei file, gestiti dal linguaggio BASIC, che si servono di supporti fisici del tipo 3, cassette magnetiche, e del tipo 4, floppy disk (dischetti).

Caratteristica di queste registrazioni è che si mantengono nel tempo a meno che non si intervenga su di esse in modo distruttivo.
Non consideriamo file le registrazioni nella memoria di lavoro del calcolatore, e, in particolare, nella memoria dedicata al video, in quanto hanno la caratteristica di essere temporanee. Abbiamo, però, visto nel primo volume che anche il video può essere gestito come un file.
È ovvio che tutto quello che diventa registrazione su un file, qualunque sia il supporto, transita dalla memoria del calcolatore.

I dati, presenti nella memoria del calcolatore, sono identificabili per mezzo dei nomi delle variabili che li contengono.

Le operazioni di SCRITTURA SU FILE da programma usano i nomi delle variabili per far uscire i dati o per prepararli ad uscire.

I dati, letti dall'esterno nella memoria del calcolatore, sono identificabili per mezzo dei nomi delle variabili che li ricevono. Le operazioni di LETTURA DA FILE usano i nomi delle variabili per far entrare i dati o per renderli disponibili dopo che sono entrati in memoria.

Gli elementi in gioco in questo tipo di operazioni sono quattro:

A) le VARIABILI che contengono o conterranno i dati,

B) il BUFFER, una zona di memoria di raccolta e transito per i dati,

C) il SUPPORTO fisico che contiene o conterrà le registrazioni,

D) il SOFTWARE DI GESTIONE.

Abbiamo già visto, nel volume dedicato al BASIC, come viene gestito il BUFFER della stampante. Riepiloghiamo qui brevemente l'argomento:

- nella stampante è contenuta una zona di memoria RAM che viene usata come buffer,
- il programma manda dati alla stampante servendosi dei nomi delle variabili,
- il buffer via via si riempie,
- il buffer si svuota, cioè avviene fisicamente la stampa, quando è pieno oppure quando il programma invia un apposito carattere di controllo o chiude la trasmissione,
- il calcolatore invia i dati alla stampante carattere per carattere in codice ASCII,
- il software di gestione per l'invio dei dati sta nella ROM del calcolatore,
- il software necessario a trasformare i codici presenti nel buffer in caratteri stampati sta nella ROM della stampante.

Cose analoghe succedono con le periferiche, argomento della presente trattazione.

Per il registratore a nastro il BUFFER sta nella memoria RAM del calcolatore e il software necessario alla trasmissione sta nella ROM del calcolatore.

Il programma SCRIVE il contenuto delle variabili, i dati si ammucchiano nel buffer, il buffer viene fisicamente trasferito sul nastro (che si mette in movimento prima dell'inizio dell'operazione) quando esso è pieno, oppure quando il programma chiude la trasmissione. Il programma LEGGE i dati nelle variabili prelevandoli dal buffer, il buffer viene riempito alla prima lettura e tutte le volte che si cerca di leggere un carattere, dopo l'ultimo che esso contiene.

Le operazioni di lettura e scrittura del nastro avvengono in modo trasparente per l'utente; egli si accorge che sta avvenendo fisicamente un'operazione solo se guarda il nastro e lo vede in movimento.

Per l'unità a dischi, invece, il sistema operativo del calcolatore provvede solo all'invio dei dati o alla loro ricezione carattere per carattere (byte a byte) e alla gestione di una tabella relativa ai file aperti. Tale tabella contiene informazioni su tutti i file aperti sulle periferiche collegate al calcolatore. Il buffer e il software necessario alla gestione dei dischetti stanno rispettivamente nella RAM e nella ROM dell'unità a dischi. L'utente non si accorge quando i dati viaggiano tra il calcolatore e l'unità a dischi, mentre si accorge che sta avvenendo fisicamente un'operazione di lettura o scrittura dal fatto che si accende la spia rossa luminosa sull'unità. Il software necessario alla gestione delle operazioni disco prende il nome di DOS (Disk Operating System).

Riepiloghiamo nella figura che segue le relazioni che legano gli elementi sopra discussi.
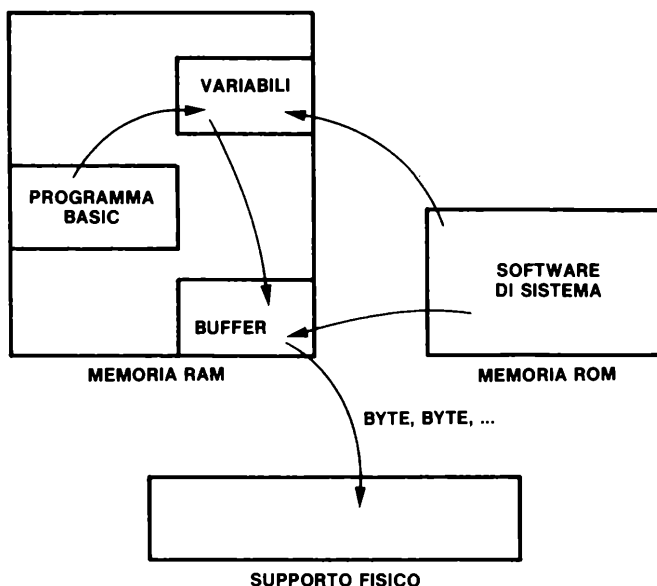
Figura 1.1 SOFTWARE, VARIABILI, BUFFER, SUPPORTO

In un file dobbiamo sempre distinguere l'aspetto fisico da quello logico. La STRUTTURA FISICA di un file dipende dal supporto di registrazione, dalla periferica e dal software di gestione a livello di linguaggio macchina (SOFTWARE DI BASE). La STRUTTURA LOGICA, che si sovrappone a una definita struttura fisica, dipende dal programma che crea il file. La struttura fisica in generale consiste nel fatto di potere e dovere scrivere un blocco fisico di byte di certe dimensioni. Per BLOCCO FISICO si intende la quantità di byte che viene fisicamente trasferita tra supporto e calcolatore, o viceversa, in una operazione di Input o Output. Per BLOCCO LOGICO si intende, invece, quel gruppo di dati che interessa il programma in un certo momento; il blocco logico può coincidere con un blocco fisico, costituirne una parte, o anche essere formato da più di un blocco fisico.

Vediamo ora la terminologia relativa all'aspetto LOGICO dei FILE. Le singole registrazioni, che si susseguono nel file e hanno un significato come gruppo di dati, si chiamano RECORD (blocco logico). In generale un record è composto da più dati, si dice che un record è composto da CAMPI (FIELD).
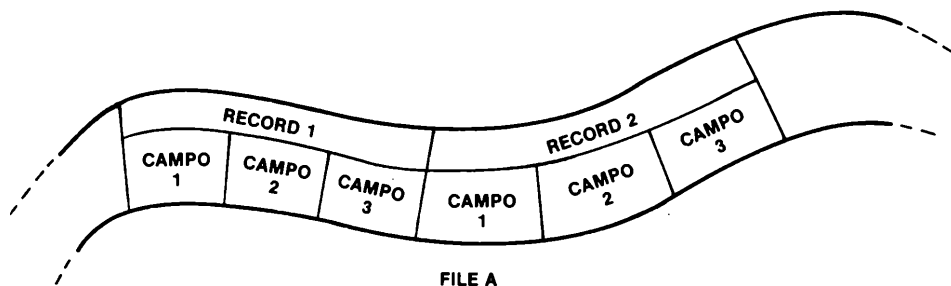
Figura 1.2 FILE, RECORD, CAMPI

Ogni campo è contraddistinto da un nome, per la logica del programma; come campo Cognome, campo Nome, campo Indirizzo, ecc.. All'interno del programma ad ogni campo corrisponde il nome di una variabile. Ogni file è contraddistinto da un nome, che lo individua. In generale al record non è attribuito un nome specifico, ma si può fare riferimento ad un record con il numero d'ordine con il quale esso compare nel file; primo record, secondo record, ecc.

Un file può essere formato da una serie di record, aventi tutti la stessa struttura, oppure da due o più strutture diverse di record che si ripetono con un certo ordine. Inoltre i record possono essere tutti della stessa lunghezza oppure di lunghezza variabile. I record di lunghezza fissa sono in generale formati da campi di lunghezza fissa. I record di lunghezza variabile possono essere formati da campi di lunghezza variabile, oppure da campi di lunghezza fissa, ma che si ripetono in numero diverso.

Nel primo volume, a proposito della stampante, non si è esplicitamente parlato di file. Vediamo ora come si può applicare la terminologia dei file alla stampa su carta.

Un qualunque tabulato è un file; ogni linea di stampa può essere considerata un record logico. Una riga di stampa (o il contenuto del buffer) può essere vista come un blocco fisico. Nella lista su carta di un programma BASIC ogni linea di programma può essere considerata un record logico di lunghezza variabile, formato da campi di lunghezza variabile. In un elenco ordinato di nomi e indirizzi ogni argomento può essere pensato come un record formato dai campi: cognome, nome, indirizzo, ecc.. Qualora il prospetto venga stampato ponendo sulla prima linea il cognome e nome e sulla seconda l'indirizzo, si può pensare a due tipi record che si ripetono in coppia.

Per ogni file si definisce il TIPO e il METODO DI ACCESSO.

Si definisce il TIPO di un file in dipendenza dal modo come il file viene costruito. Noi ci occupiamo dei tipi di file consentiti per il COMMODORE 64, in BASIC, e precisamente dei:

- File SEQUENZIALI,
- File RANDOM,
- File RELATIVI.

Un file può consistere in un insieme di dati, tipo gli elementi di un vettore di numeri, registrati uno dopo l'altro. Si tratta di una lista di dati e il file viene definito di TIPO SEQUENZIALE. Nell'esempio riportato ogni numero è un record del file, quindi il record è costituito da un solo campo. Caratteristica del file sequenziale è che ogni record, salvo il primo e l'ultimo, è preceduto e seguito da un altro record. Il file sequenziale si costruisce cominciando a scrivere il primo record e procedendo in sequenza.

Abbiamo un file di TIPO RANDOM quando i record vengono scritti facendo riferimento all'indirizzo del blocco fisico dove sono registrati. In realtà la terminologia, adottata nella letteratura della casa costruttrice, che nel seguito useremo anche noi, può NON DARE una chiara visione di questo tipo di file. A noi sembra che la definizione più corretta sia: FILE AD ACCESSO DIRETTO PER MEZZO DELL'INDIRIZZO FISICO. Approfondiremo l'argomento nel Capitolo 3, chiarendo la relazione che sussiste, in questo caso, tra record logico e blocco fisico. I file random si costruiscono scrivendo i record come si vuole (nel rispetto delle regole del particolare file che si tratta), ma senza l'obbligo della sequenzialità, nel senso visto sopra.

Abbiamo un file DI TIPO RELATIVO quando i record sono scritti citando il loro numero d'ordine nel file: primo, secondo, quinto, ventesimo, terzo, ecc., senza l'obbligo di rispettare un ordine sequenziale. Riprendendo la definizione precedente possiamo definire questo tipo come: FILE AD ACCESSO DIRETTO PER MEZZO DELL'INDIRIZZO LOGICO.

Il tipo di un file viene definito al momento della sua creazione e non è modificabile.

Per METODO DI ACCESSO si intende il modo nel quale si può accedere ai record che compongono il file. I metodi di accesso disponibili sono due:

- Accesso SEQUENZIALE,

- Accesso DIRETTO.

Il metodo di accesso sequenziale consiste nella lettura dei record uno dopo l'altro, partendo dal primo; esso si può usare per tutti i tre tipi di file sopra descritti. Il metodo di accesso diretto, che consiste nella lettura di un ben determinato record, indipendentemente dalla lettura precedente, si può usare solo per i file random e per i file relativi.
Sulla cassetta è possibile solo la gestione dei file sequenziali. Sui dischetti sono gestibili i tre tipi di file.

Le registrazioni magnetiche avvengono un byte alla volta, si ha la trasmissione di sequenze di byte. Ogni byte contiene un codice numerico che può variare da 0 a 255, e che viene abitualmente definito carattere. Anche i dati numerici, che all'interno della memoria del calcolatore sono memorizzati come numeri interi o floating-point, sono trasformati in uscita in stringhe di caratteri. Una trasformazione inversa avviene in fase di lettura dal supporto magnetico verso la memoria del calcolatore. Ogni stringa di caratteri, che rappresenta un dato, deve terminare con un carattere separatore riconoscibile dal sistema.

1.2 Operazioni per la gestione dei file

Il programma BASIC per gestire un file deve svolgere le seguenti operazioni:

1) Aprire la comunicazione con il file,

2) Leggere o scrivere il file,

3) Chiudere la comunicazione con il file.

L'operazione 1) stabilisce la comunicazione con la periferica e con un determinato file, per effettuare un certo tipo di operazione. Si tratta dell'operazione OPEN; la parola chiave è seguita da alcuni parametri, che analizziamo separatamente per la cassetta e per il dischetto nei due capitoli seguenti. Dopo la OPEN i dati relativi al file aperto sono memorizzati nella RAM del calcolatore nella tabella per la gestione dei file. Il byte 152 (0098H) contiene il numero dei file aperti (LDTND), puntatore alla tabella dei file. La tabella di gestione dei file si trova a partire dal byte 601 (0259H); essa è formata da tre vettori di 10 byte ciascuno. La struttura è la seguente:

- LAT da 601 a 610 (0259H-0262H) per i numeri logici dei file attivi.
- FAT da 611 a 620 (0263H-026CH) per i numeri logici delle periferiche dei file attivi.
- SAT da 621 a 630 (026DH-0276H) per gli indirizzi secondari dei file attivi.

I tre byte 601, 611 e 621 contengono le informazioni relative al primo file, e gli altri, analogamente in gruppi di tre, per gli altri file. Dalle dimensioni della tabella sembra che si possano aprire contemporaneamente 10 file. In realtà vedremo che non è così.

L'operazione 2) legge o scrive gruppi di dati trasferendoli dal o nel buffer e, certe volte, dà luogo anche alla lettura o scrittura fisica nel o dal buffer.
Anche questa operazione viene trattata diffusamente nei due capitoli seguenti; sono le istruzioni INPUT #, GET# e PRINT #.

L'operazione 3) chiude la comunicazione con il file e la periferica; si tratta della CLOSE. Essa elimina il file dalla tabella di gestione, e, in certi casi, dà luogo all'ultima registrazione sul supporto.

1.3 Come si elaborano i file di dati

Le operazioni che si eseguono sui file di dati possono essere dei tipi seguenti:

1) Creazione ex-novo del file.
2) Aggiornamento del file.
3) Ricerca di dati sul file.

Il modo con il quale possono avvenire queste operazioni dipende dal tipo del file, definito nella fase di creazione.

CREAZIONE EX-NOVO DEL FILE

Consiste nella scrittura iniziale del file. In questa fase viene assegnato un nome al file, ne viene stabilito il tipo, sono definite le caratteristiche dei record e dei campi.

Nei due capitoli seguenti precisiamo come viene svolta la creazione del file per ogni tipo preso in esame.

AGGIORNAMENTO DEL FILE

Consiste nella modifica di record già presenti nel file, nella cancellazione di qualche record e nell'aggiunta di nuovi record. L'operazione si svolge con modalità diverse in dipendenza dal tipo del file e dal supporto fisico usato.

RICERCA DI DATI SUL FILE

Possiamo raggruppare sotto questo nome tutte le operazioni di utilizzo dei dati registrati in un file. La più semplice e banale è la lista su carta di tutti i record del file. Altre operazioni possono essere di tipo selettivo con riferimento al contenuto di uno specifico campo. Un'operazione molto comune è la riorganizzazione dei record di un file in base al contenuto di un campo; essa prende il nome di SELEZIONE (SORTING) e richiede una notevole quantità di memoria, sia centrale che di massa. Le modalità di ricerca consentite su un file dipendono dal suo tipo, dal supporto fisico e dal software a disposizione.

Il problema principale inerente alla ricerca di un record in un file è quello di sapere quale record si vuole. Consideriamo, come esempio, un file relativo ad alcune informazioni sui dodici mesi dell'anno. Questo file è formato da 12 record, uno per ogni mese, e risulta ovvio che il primo record è quello di gennaio, l'ultimo quello di dicembre, il decimo quello di ottobre e simili. Consideriamo ora un file di indirizzi, che contiene 500 argomenti; come facciamo a sapere che il record relativo al signor Bianchi Carlo è, per esempio, il dodicesimo? Un modo potrebbe essere avere a disposizione un elenco su carta, dove ogni nome, riportato in ordine alfabetico per comodità di consultazione, sia accompagnato da un numero, il numero d'ordine del record. Abbiamo ora introdotto il concetto di INDICE di un file, anche se su carta. Se non possediamo alcun indice, per trovare il record del signore in questione, possiamo solo cominciare a leggere il file dal primo record controllando i nomi che si incontrano, fino a trovare quello cercato. Questo metodo è, per esempio, l'unico a disposizione per i file su nastro, che sono di tipo sequenziale.

CAPITOLO 2

FILE SU CASSETTA

2.1 Introduzione

Al file su cassetta può essere assegnato un nome di riconoscimento; esso viene registrato insieme ad alcuni caratteri di controllo, generati dal sistema. L'insieme di questi dati costituisce il blocco di testata del file sul nastro. Se l'utente registra senza nome, il blocco di testata contiene solo i caratteri di controllo. Questi caratteri di controllo risultano trasparenti per l'utente nelle normali operazioni di gestione dei file da BASIC. Ti consigliamo di registrare usando sempre un nome di riconoscimento; esso permette infatti di distinguere tra loro le diverse registrazioni, che possono trovarsi sulla stessa cassetta, e di richiamarle in modo selettivo.

Il COMMODORE 64 consente di gestire due tipi di file su nastro; si tratta sempre di FILE SEQUENZIALI, ma le modalità di registrazione sono diverse. Si possono avere:

- I file ASCII: in essi i dati sono registrati carattere per carattere in blocchi separati uno dall'altro da un CARATTERE SEPARATORE.

- I file BINARI: in essi i caratteri sono registrati in modo apparentemente continuo.

I file ASCII sono usati per registrare dati, i file BINARI per registrare programmi. I due tipi di file sono gestiti con istruzioni diverse; nel volume dedicato al BASIC abbiamo già trattato i file di programmi.

La differenza, dal punto di vista della gestione, tra i due tipi di file sta nel fatto che, mentre i file ASCII possono essere letti a pezzi e il nastro viene avviato per leggere un blocco e poi fermato, i file BINARI possono solo essere letti tutti in una sola volta; il loro contenuto viene cioè trasferito completamente in memoria a partire da

un certo byte. Inoltre, se l'indirizzo del byte dal quale inizia la memorizzazione non coincide con quello registrato nella testata del file, avviene la modifica dei due byte di LINK di ogni linea di programma (rilocazione). Questo succede in quanto i file binari possono essere solo file di programmi BASIC.

Il buffer per le operazioni nastro si trova nella RAM del calcolatore a partire dal byte 828 (033CH) e occupa 192 byte, in conseguenza il blocco fisico di nastro che viene letto o scritto è di 192 caratteri. Il sistema registra due volte ogni blocco fisico su nastro e, in fase di lettura, confronta la prima lettura con la seconda. Se le due letture non coincidono viene segnalato un errore. L'utente non si accorge di questa doppia registrazione, può solo notare che le operazioni nastro sono lente. Tra un blocco fisico di registrazione e il successivo si trova un pezzetto di nastro non utilizzato (GAP); esso è necessario per consentire la fermata e il riavvio del nastro. Ogni file è composto almeno da tre blocchi fisici: la testata, il corpo del file e il blocco di chiusura.

Altri indirizzi importanti per la gestione delle operazioni su cassetta sono:

FBUFPT, byte 113/114 (0071H-0072H)
puntatore al buffer della cassetta.

STATUS, byte 144 (0090H)
parola di stato per le routine KERNAL di Input/Output, nel programma BASIC viene richiamata con ST.

SVXT, byte 146 (0092H)
costante di controllo per la velocità di scorrimento del nastro.

VERCK, byte 147 (0093H)
indicatore: 0=LOAD, 1=VERIFY.

SYNO, byte 150 (0096H)
numero per sincronizzazione cassetta.

RTY, byte 155 (009BH)
carattere di parità del nastro.

DPSW, byte 156 (009CH)
indicatore ricezione byte da nastro.

PTR1, byte 158 (009EH)
indicatore errore passo 1 (primo blocco) nastro.

PTR2, byte 159 (009FH)
indicatore errore passo 2 (secondo blocco) nastro.

CNTDN, byte 165 (00A5H)
contatore in decremento per sincronizzazione cassetta.

BUFPNT, byte 166 (00A6H)
puntatore al buffer del nastro.

INBIT, byte 167 (00A7H)
bit di input cassetta.

BITC1, byte 168 (00A8H)
contatore bit input cassetta.

RIDATA, byte 170 (00AAH)
byte di input cassetta.

RIPRTY, byte 171 (00ABH)
contatore corto cassetta.

SAL, byte 172/173 (00ACH-00ADH)
indirizzo iniziale memorizzazione file binari.

EAL, byte 174/175 (00AEH-00AFH)
indirizzo finale memorizzazione file binari.

CMPO, byte 176/177 (00B0H-00B1H)
costanti per operazioni nastro.

TAPE1, byte 178/179 (00B2H-00B3H)
puntatore buffer nastro.

BITTS, byte 180 (00B4H)
contatore bit output cassetta.

NXTBIT, byte 181 (00B5H)
prossimo bit da inviare, indicatore fine nastro.

FNLEN, byte 183 (00B7H)
lunghezza nome file corrente.

LA, byte 184 (00B8H)
numero logico file corrente.

SA, byte 185 (00B9H)
indirizzo secondario file corrente.

FA, byte 186 (00BAH)
numero logico periferica corrente.

FNADR, byte 187/188 (00BBH-00BCH)
puntatore al nome del file corrente.

ROPRTY, byte 189 (00BDH)
parità output cassetta.

FSBLK, byte 190 (00BEH)
contatore blocco Input/Output cassetta.

CAS1, byte 192 (00C0H)
arresto motore nastro.

MEMUSS, byte 195/196 (00C3H-00C4H)
memoria temporanea LOAD nastro.

IRQTMP, byte 671/672 (029FH-02A0H)
contiene il vettore IRQ durante Input/Output da nastro.

Byte 674 (02A2H)
controllo sensore cassetta durante Input/Output da nastro.

Byte 675 (02A3H)
memoria temporanea per lettura nastro.

Byte 676 (02A4H)
indicatore temporaneo IRQ durante lettura nastro.

Puoi analizzare il contenuto di alcuni di questi byte in diversi momenti, usando la funzione PEEK, per renderti conto meglio del loro significato.

2.2 File di dati

Su nastro si possono gestire solo file sequenziali. Il file può essere composto da record di lunghezza fissa o variabile. Il record può essere composto da un solo campo o da più campi di lunghezza fissa o variabile. Il fatto di gestire campi di lunghezza variabile (e in conseguenza record di lunghezza variabile), dipende dalla logica del programma. Per la logica di costruzione del file un campo termina con un carattere separatore valido, ma al limite, e con le limitazioni che vedremo, un intero file potrebbe essere costituito da un solo campo senza caratteri separatori.

Esponiamo per prima cosa le istruzioni BASIC disponibili per la gestione dei file su cassetta, facendo seguire alcuni esempi.

OPEN lfn,dn,sa,fn

si deve usare per aprire un file su nastro, ponendo: dn=1 o omettendolo, dato che viene assunto per default uguale a uno. Per omettere un parametro, qualora ne seguano altri, si devono scrivere due virgole vicine.

Il parametro sa può avere i seguenti valori:

sa=0 o mancante (valore di default zero) per operazione di lettura,
sa=1 per operazione di scrittura senza registrazione del carattere di fine-nastro (END-OF-TAPE) dopo la chiusura del file,
sa=2 per operazioni di scrittura con registrazione del carattere di fine-nastro dopo la chiusura del file.

Il parametro lfn (logical-file-number) può avere un valore compreso tra 1 e 255, ma di solito si usano numeri inferiori a 127; tale numero deve essere sempre citato nelle istruzioni relative al file.
Il nome del file, fn, deve essere una stringa di al massimo 16 caratteri; esso può essere passato come costante o come variabile. Tale nome può essere omesso, ma la cosa non è consigliabile.
Tutti gli altri parametri che compaiono nell'istruzione possono essere costanti o variabili numeriche.

L'istruzione OPEN provoca la scrittura del blocco di testata del file, tale blocco transita dal buffer, come vedremo dagli esempi che seguono.

PRINT#lfn,lista

provoca la scrittura dei dati della lista sul file contraddistinto dal numero logico lfn. Nel caso della cassetta può essere ovviamente trattato un file per volta, ma se lo lfn

usato nella PRINT non corrisponde a quello usato nella OPEN si ha segnalazione di errore. Analogamente si ha segnalazione di errore se il file non è stato aperto per scrivere. I dati della lista vengono trasferiti nel buffer della cassetta; quando il buffer è pieno, esso viene scritto su nastro. Può succedere che l'ultimo dato inviato, quello che finisce di riempire il buffer, venga spezzato in modo che i primi caratteri sono scritti fisicamente in un blocco e gli altri nel successivo. Tu comunque non ti accorgi di questo nè in fase di scrittura, nè in fase di lettura.

Come sempre, la lista di dati può essere formata da costanti o da variabili separate tra loro da virgola o da punto e virgola. Il carattere separatore virgola provoca l'aggiunta di spazi ai caratteri del dato che lo precede, il carattere separatore punto e virgola non provoca l'aggiunta di spazi. In sostanza questi caratteri separatori agiscono come sul video, ma non provocano la registrazione di un carattere separatore. Per poter rileggere poi con l'istruzione INPUT#, che legge in variabili, è necessario che i dati siano registrati facendo comparire dopo ognuno di essi un carattere separatore valido, cioè riconosciuto come tale dal sistema. I caratteri separatori riconosciuti sono il RETURN (CHR$(13)) e la virgola (CHR$(44)). Il primo può essere passato usando la funzione CHR$(13) oppure facendo terminare la lista di dati senza punteggiatura. Il secondo può essere passato usando la funzione CHR$(44) oppure come stringa costante ",". Abbiamo però delle limitazioni nell'uso dei due caratteri separatori, infatti il sistema non consente che i dati, qualora debbano poi essere letti con INPUT#, siano registrati in modo che tra due caratteri CHR$(13) siano compresi più di 79 caratteri. Inoltre, se si usa come carattere separatore registrato la virgola, l'istruzione INPUT# deve leggere riportando nella lista dei dati tanti nomi di variabili quanti sono i dati compresi tra due CHR$(13); se non si opera così, si perdono dei dati.

In sostanza possiamo affermare che:

- per separare i record logici si deve usare il carattere separatore CHR$(13), mentre per separare i campi si può usare anche il carattere CHR$(44), purchè la somma dei caratteri non superi 79, comprese le virgole separatrici, e i campi vengano letti tutti da una sola INPUT#;
- per poter leggere con INPUT# il numero dei caratteri compresi tra due CHR$(13) non deve superare 79;
- quando la lista dei dati termina senza punteggiatura il sistema aggiunge un CHR$(13);
- può essere usato sempre come carattere separatore CHR$(13) sia a livello di campo che di record, e con esso si può leggere con maggior libertà;
- i dati vengono scritti carattere per carattere trasformando i numeri in stringa con le regole note;

• i dati, scritti senza caratteri separatori registrati o con caratteri separatori registrati, possono essere letti carattere per carattere con l'istruzione GET#:

INPUT#lfn,lista

legge dal file, aperto per leggere con numero logico lfn, i dati trasferendoli nelle variabili della lista. Un dato viene considerato terminato quando viene incontrato il carattere CHR$(13) o il carattere CHR$(44). Se un dato supera il numero dei caratteri consentiti si ha segnalazione di errore. Se si cerca di leggere in una variabile numerica un dato non numerico si ha segnalazione di errore. Se la lista contiene un numero di variabili inferiore al numero dei dati compresi tra due RETURN e separati da virgole, i dati in più non vengono letti.

GET#lfn,lista

legge dal file, aperto per leggere con numero logico lfn, un carattere per volta. Dopo l'operazione ogni variabile della lista contiene un carattere. Ti consigliamo di usare nella lista solo variabili stringa, per evitare errori.
Con GET# si può leggere un file scritto senza usare mai caratteri separatori validi tra i dati; deve, però, pensare poi il programma a interpretare opportunamente l'insieme dei caratteri letti.

CLOSE lfn

chiude il file aperto con numero logico lfn.
Segue il programma PRFL1C, come primo esempio.

```
10 REM PRFL1C
15 REM CREAZIONE FILE SEQUENZIALE
20 REM RECORD LOGICO FORMATO DA 3 CAMPI
25 REM CAMPI DI LUNGHEZZA VARIABILE
30 REM A$←COGNOME
35 REM B$←NOME
40 REM C$←DATO NUMERICO LETTO COME STRINGA
45 REM SEPARATORE DI CAMPO = CHR$(13)
50 REM UN SOLO PRINT SCRIVE IL RECORD
55 REM SENZA PUNTEGGIATURA FINALE
60 REM IL SISTEMA AGGIUNGE CHR$(13)
65 REM ALLA FINE DEL RECORD
70 REM ";" COME SEPARATORE TRA LE VARIABILI
75 REM PER NON AGGIUNGERE SPAZI
80 CH$=CHR$(13):OPEN1,1,1,"FL1C"
```

```
85 V$=CHR$(44):PV$=CHR$(59)
90 CD$="SQQQQQQQQQQQQQQQQQQQQQQQ"
95 PRINT"♥":INPUT"COGNOME:  ";A$
100 INPUT"NOME:  ";B$
105 INPUT"NUMERO:";C$
110 PRINT#1,A$;CH$;B$;CH$;C$
115 PRINT#1,A$,CH$,B$,CH$,C$
120 PRINT#1,A$;V$;B$;V$;C$
125 PRINT#1,A$;PV$;B$;PV$;C$
130 R$="":PRINT CD$;"ALTRO RECORD (S/N)?";
135 GETR$:IFR$=""THEN135
140 IFR$<>"S"THEN150
145 GOTO95
150 PRINT"♥RIAVV. IL NASTRO-POI SCRIVI CONT"
155 CLOSE1:STOP
160 OPEN1,1,0,"FL1C":OPEN4,4
165 PRINT#4,"FILE LETTO CON GET#":PRINT#4
170 GET#1,R$:TS=ST:PRINT#4,R$;"*";
175 IFTS<>64THEN170
180 PRINT#4:CLOSE1:CLOSE4
185 PRINT"♥RIAVV. IL NASTRO-POI SCRIVI CONT"
190 STOP
195 OPEN1,1,0,"FL1C":OPEN4,4
200 PRINT#4,"FILE LETTO CON INPUT#":PRINT#4
205 INPUT#1,R$:TS=ST:PRINT#4,R$
210 IFTS<>64THEN205
215 PRINT#4:CLOSE1:CLOSE4:STOP
```

Il programma PRFL1C fa le seguenti cose:

- crea un file sequenziale su nastro di nome FL1C (linea 80),
- il record logico del file è formato da 3 campi di lunghezza variabile, e per essi non viene controllata la lunghezza: cognome, nome e numero (linee 95-105),
- i 3 campi sono trattati come stringhe,
- ogni record logico viene scritto 4 volte:

1).usando come carattere separatore tra le variabili il punto e virgola e come carattere separatore registrato il CHR$(13) (linea 110),

2).come in 1), ma usando come carattere separatore tra le variabili la virgola (linea 115),

3).usando come carattere separatore tra le variabili il punto e virgola e come carattere separatore registrato la virgola (CHR$(44)) (linea 120),

4).come in 3), ma usando come carattere separatore registrato il punto e virgola (CHR$(59)) (linea 125),

- dopo aver scritto 4 volte un record viene chiesto se si vuole aggiungere un altro record (un gruppo di 4) (linee 130-145),
- quando la risposta è NO, viene chiuso il file (linea 155),
- viene chiesto di riavvolgere il nastro (linea 150) e viene aperto il file per lettura (linea 160),
- il file viene letto con GET# e stampato carattere per carattere (linee 170-180),
- viene chiesto di riavvolgere il nastro (linea 185) e viene aperto nuovamente il file per lettura (linea 195),
- il file viene letto con INPUT# e stampato (linee 200-215).

Seguono i risultati di FL1C, nel caso si siano caricati 2 record diversi, contenenti i seguenti dati:

Cognome: PRIMOC, Nome: PRIMON, Numero: 11111
Cognome: SECONDOC, Nome: SECONDON, Numero: 22222

```
FILE LETTO CON GET#

P*R*I*M*O*C*
*P*R*I*M*O*N*
*1*1*1*1*1*
*P*R*I*M*O*C* * * * * * * * * * *
* * * * * * * * * * *P*R*I*M*O*N* * * * * * * * * * *
* * * * * * * * * * *1*1*1*1*1*
*P*R*I*M*O*C*,*P*R*I*M*O*N*,*1*1*1*1*1*
*P*R*I*M*O*C*;*P*R*I*M*O*N*;*1*1*1*1*1*
*S*E*C*O*N*D*O*C*
*S*E*C*O*N*D*O*N*
*2*2*2*2*2*
*S*E*C*O*N*D*O*C* * * * * * * * * * *
* * * * * * * * * * *S*E*C*O*N*D*O*N* * * * * * * * * * *
* * * * * * * * * * *2*2*2*2*2*
*S*E*C*O*N*D*O*C*,*S*E*C*O*N*D*O*N*,*2*2*2*2*2*
*S*E*C*O*N*D*O*C*;*S*E*C*O*N*D*O*N*;*2*2*2*2*2*
*
```

```
FILE LETTO CON INPUT#

PRIMOC
PRIMON
11111
PRIMOC
PRIMON
11111
PRIMOC
PRIMOC;PRIMON;11111
SECONDOC
SECONDON
22222
SECONDOC
SECONDON
22222
SECONDOC
SECONDOC;SECONDON;22222
```

Per mettere meglio in evidenza i risultati della lettura con GET#, ogni carattere viene stampato facendolo seguire dal carattere "*". Come puoi vedere:

- il carattere separatore ";" tra le variabili non aggiunge spazi,
- il carattere separatore "," tra le variabili aggiunge 10 spazi dopo il dato,
- dopo il CH$ (che è il carattere RETURN), la stampa continua alla riga seguente e ci sono 10 spazi, il primo asterisco è quello che viene dopo CH$,
- i caratteri separatori registrati "," e ";" sono letti e stampati,
- ogni istruzione di PRINT# (linee 110-125) termina senza punteggiatura e quindi il sistema aggiunge il RETURN.

La seconda parte dei risultati riguarda la lettura con INPUT#. Come puoi vedere:

- non si nota differenza nella lettura dei primi due record, infatti gli spazi a sinistra dei dati vengono persi e quelli a destra non si vedono (ma ci sono),
- per il terzo record la virgola separatrice registrata fa perdere gli ultimi due campi, dato che la lista dati della INPUT# contiene una sola variabile,
- per il quarto record il punto e virgola separatore registrato non viene riconosciuto e i tre campi vengono letti come un campo unico.

Puoi sostituire alla linea 205 del programma le due linee che seguono:

```
205 INPUT#1,R1$,R2$,R3$:TS=ST
206 PRINT#4,R1$:PRINT#4,R2$:PRINT#4,R3$
```

ottenendo di non perdere i tre campi quando usi il separatore registrato virgola. Infatti in questo caso la linea 205 legge tutto il record in una sola volta; la lista dati è formata da tre variabili.

Puoi provare il programma PRFL1C rispondendo alla richiesta dati con campi abbastanza lunghi, tipo 60 caratteri; vedrai che alla INPUT# di linea 205 otterrai il messaggio di errore STRING TOO LONG, infatti con l'aggiunta degli spazi si superano i 79 caratteri.

Segue il programma PRFL2C.

```
10 REM PRFL2C
15 REM CARICAMENTO FILE NUMERICO
20 REM OGNI RECORD LOGICO COMPRENDE 2 CAMPI
25 REM IL PRIMO E' UN NUMERO INTERO
30 REM IL SECONDO UN NUMERO DECIMALE
35 REM SI USA CHR$(13) COME SEPARATORE DI CAMPO
40 REM PER USCIRE DARE UN INTERO NULLO
45 REM IL RETURN FINALE LO METTE IL SISTEMA
50 CH$=CHR$(13)
55 OPEN1,1,1,"FL2C":OPEN4,4
60 PRINT"▣NUMERI INTERI <= 32767▣▣"
65 PRINT"PER USCIRE RISPONDERE 0 (ZERO) A INTERO▣▣"
70 INPUT"INTERO: ";I%
75 IFI%=0THENCLOSE1:GOTO95
80 INPUT"DECIMALE: ";D
85 PRINT#1,I%;CH$;D
90 GOTO70
95 Z$="DOPO OPERAZIONE CLOSE":GOSUB185
100 PRINT"▣RIAVV. IL NASTRO-POI SCRIVI CONT"
105 STOP
110 REM LETTURA FILE NUMERICO
115 REM CON INPUT VENGONO LETTI I 2 CAMPI
120 REM ESSI VENGONO STAMPATI
```

```
125 REM QUANDO E' FINITO IL FILE VIENE LETTO
130 REM E STAMPATO IL CONTENUTO DEL BUFFER
135 OPEN1,1,0,"FL2C"
140 Z$="DOPO OPERAZIONE OPEN":GOSUB185
145 PRINT#4:PRINT#4,"CONTENUTO FILE NUMERICO"
150 PRINT#4:PRINT#4
155 INPUT#1,I%,D:TS=ST
160 PRINT#4,I%,D
165 IFTS<>64THEN155
170 Z$="DOPO LETTURA DATI NUMERICI":GOSUB185
175 CLOSE1
180 CLOSE4:STOP
185 PRINT#4:PRINT#4,Z$
190 PRINT#4,"CONTENUTO BUFFER CASSETTA"
195 PRINT#4
200 I=828
205 FORK=1TO18:FORL=1TO4
210 PRINT#4,I;" ";MID$(STR$(PEEK(I))+"   ",2,3);"
215 I=I+1:NEXTL:PRINT#4
220 NEXTK:PRINT#4:RETURN
```

Il programma PRFL2C fa le seguenti cose:

- crea un file sequenziale il cui record logico comprende due campi numerici: un numero intero e un numero decimale (con cifre dopo il punto decimale) (linee 55-90),
- viene usato come carattere separatore sia di campo che di record il RETURN (linea 85),
- dopo la chiusura del file viene stampato il contenuto del buffer della cassetta (linea 95, sottoprogramma 185-220),
- viene chiesto di riavvolgere il nastro (linea 100) e poi viene aperto il file per leggere (linea 135),
- dopo l'apertura del file viene stampato il contenuto del bufffer (linea 140),
- viene letto e stampato il file (linee 145-165),
- viene stampato il contenuto del buffer dopo la lettura del file (linea 170).

Seguono i risultati nel caso si siano caricate nel file le seguenti 4 coppie di numeri:

123, -678.43
-67, 543.78
3456, 90.67
-43, 890.5

```
DOPO OPERAZIONE CLOSE
CONTENUTO BUFFER CASSETTA

 828  2     829  32    830  49    831  50
 832  51    833  32    834  13    835  45
 836  54    837  55    838  56    839  46
 840  52    841  51    842  32    843  13
 844  45    845  54    846  55    847  32
 848  13    849  32    850  53    851  52
 852  51    853  46    854  55    855  56
 856  32    857  13    858  32    859  51
 860  52    861  53    862  54    863  32
 864  13    865  32    866  57    867  48
 868  46    869  54    870  55    871  32
 872  13    873  45    874  52    875  51
 876  32    877  13    878  32    879  56
 880  57    881  48    882  46    883  53
 884  32    885  13    886  0     887  32
 888  32    889  32    890  32    891  32
 892  32    893  32    894  32    895  32
 896  32    897  32    898  32    899  32
```

```
DOPO OPERAZIONE OPEN
CONTENUTO BUFFER CASSETTA

 828  4     829  60    830  3     831  80
 832  12    833  70    834  76    835  50
 836  67    837  32    838  32    839  32
 840  32    841  32    842  32    843  32
```

```
844   32    845   32    846   32    847   32
848   32    849   32    850   32    851   32
852   32    853   32    854   32    855   32
856   32    857   32    858   32    859   32
860   32    861   32    862   32    863   32
864   32    865   32    866   32    867   32
868   32    869   32    870   32    871   32
872   32    873   32    874   32    875   32
876   32    877   32    878   32    879   32
880   32    881   32    882   32    883   32
884   32    885   32    886   32    887   32
888   32    889   32    890   32    891   32
892   32    893   32    894   32    895   32
896   32    897   32    898   32    899   32
```

```
CONTENUTO FILE NUMERICO


 123            -678.43
-67             543.78
 3456             90.67
-43             890.5

DOPO LETTURA DATI NUMERICI
CONTENUTO BUFFER CASSETTA
```

```
828   2     829   32    830   49    831   50
832   51    833   32    834   13    835   45
836   54    837   55    838   56    839   46
840   52    841   51    842   32    843   13
844   45    845   54    846   55    847   32
848   13    849   32    850   53    851   52
852   51    853   46    854   55    855   56
856   32    857   13    858   32    859   51
860   52    861   53    862   54    863   32
864   13    865   32    866   57    867   48
868   46    869   54    870   55    871   32
872   13    873   45    874   52    875   51
876   32    877   13    878   32    879   56
880   57    881   48    882   46    883   53
```

```
884  32   885  13   886  0    887  32
888  32   889  32   890  32   891  32
892  32   893  32   894  32   895  32
896  32   897  32   898  32   899  32
```

Come puoi vedere, il contenuto del buffer dopo la CLOSE che chiude la creazione del file è uguale a quello stampato dopo la lettura del file. Trovi come primo carattere 2, che significa file di dati. Dopo trovi i numeri caricati, carattere per carattere in codice ASCII, preceduti da uno spazio (32) o dal segno meno (45), dopo il numero si ha ancora uno spazio (32) e poi il carattere separatore RETURN (13). Dopo l'ultimo numero, nel byte 886 puoi vedere lo zero binario di chiusura del file. Il contenuto del buffer dopo l'operazione di OPEN per leggere ci mostra 4 nel primo byte, e il nome del file FL2C nei byte da 833 a 836.

2.3 File di programmi

Nel Paragrafo 4.2 del volume IL BASIC, abbiamo visto i comandi relativi alla memorizzazione e al caricamento dei programmi su nastro. In questo paragrafo analizziamo il contenuto dei file di programmi, cercando di leggerli come file sequenziali ASCII.

Abbiamo preso il programma PRGINMEM di pag. 267 del primo volume e l'abbiamo salvato su cassetta con: SAVE "PRGINMEM".
Il programma FILEPROG1, che segue, legge tale file aprendolo come file sequenziale ASCII.

```
10 REM FILEPROG1
15 GOSUB145:REM AZZERA BUFFER
20 OPEN1,1,0,"PRGINMEM"
25 OPEN4,4
30 PRINT#4,"BUFFER DOPO OPEN PRGINMEM"
35 PRINT#4:GOSUB100:REM LISTA BUFFER
40 GOSUB145:REM AZZERA BUFFER
45 GET#1,A$:REM LEGGE UN CARATTERE
50 PRINT#4:PRINT#4,"BUFFER DOPO UN GET"
55 PRINT#4:GOSUB100:REM LISTA BUFFER
```

```
60 REM CICLO 191 GET PER ARRIVARE ALLA
65 REM FINE DEL BUFFER
70 FORK=1TO191:GET#1,A$
75 NEXTK
80 GET#1,A$:REM UN GET DOPO I PRIMI 192
85 PRINT#4:PRINT#4,"BUFFER DOPO 193 GET"
90 PRINT#4:GOSUB100:REM LISTA BUFFER
95 CLOSE1:CLOSE4:STOP
100 REM LISTA BUFFCASS
105 REM INDIRIZZI BUFFER CASSETTA
110 I1=828:I2=1019
115 FORI=I1TOI2STEP6
120 FORK=0TO5
125 PRINT#4,PEEK(I+K);" ";
130 NEXTK
135 PRINT#4:NEXTI
140 RETURN
145 REM AZZERA BUFFCASS
150 I1=828:I2=1019
155 FORI=I1TOI2
160 POKEI,32
165 NEXTI
170 RETURN
```

Il programma FILEPROG1 fa le seguenti cose:

- pone il carattere spazio (32) in tutte le posizioni del buffer (linea 15, sottoprogramma 145-170),
- apre il file PRGINMEM come file da leggere (linea 20),
- stampa il contenuto del buffer dopo la OPEN (linee 30-35, sottoprogramma 100-140),
- pulisce il buffer (linea 40),
- legge con GET# un carattere dal file, provocando il riempimento del buffer (linea 45),
- stampa il contenuto del buffer (linee 50-55),
- esegue un ciclo di 191 GET#, per arrivare alla fine del buffer (linee 60-75),
- esegue un nuovo GET# per provocare un ulteriore riempimento del buffer (linea 80),
- stampa il contenuto del buffer dopo questo nuovo GET# (linee 85-90),
- chiude il file (linea 95).

Seguono i risultati.

```
BUFFER DOPO OPEN PRGINMEM

 1    1    8    82    9    80
 82   71   73   78   77   69
 77   32   32   32   32   32
 32   32   32   32   32   32
 32   32   32   32   32   32
 32   32   32   32   32   32
 32   32   32   32   32   32
 32   32   32   32   32   32
 32   32   32
 32   32


BUFFER DOPO UN GET

 16   8    1    0    143   32
 80   82   71   73   78   77
 69   77   0    54   8    5
 0    159  52   44   52   58
 157  52   58   153  34   80
 82   79   71   82   65   77
 77   65   32   73   78   32
 77   69   77   79   82   73
 65   34   58   153  0    76
 8    10   0    88   178  50
 53   54   172  194  40   52
 52   41   170  194  40   52
 51   41   0    98   8    15
 0    89   49   178  194  40
 88   41   58   89   50   178
 194  40   88   170  49   41
 0    129  8    17   0    139
 89   49   170  89   50   178
 48   167  153  88   59   34
 42   42   34   59   89   49
 59   89   50   58   137  54
 48   0    165  8    20   0
```

```
153   88   59   34   42   42
34    59   89   49   59   89
50    59   34   76   73   78
75    61   34   59   89   50
172   50   53   54   170  89
49    0    189  8    25   0
89    49   178  194  40   88
170   50   41   58   89   50
178   194  40   88   170  51
41    0    221  8    30   0
```

```
BUFFER DOPO 193 GET

3    1    8    82   9    80
82   71   73   78   77   69
77   32   32   32   32   32
32   32   32   32   32   32
32   32   32   32   32   32
32   32   32   32   32   32
32   32   32   32   32   32
32   32   32   32   32   32
32   32
```

Come puoi vedere dai risultati, dopo la OPEN nel buffer si trova 1 nella prima posizione. Segue nei 2 byte successivi l'indirizzo di inizio del programma (8*256+1=2049). Segue nei 2 byte successivi l'indirizzo di fine programma (9*256+82=2376). Segue il nome del file: PRGINMEM.
Il contenuto del buffer dopo il primo GET# ci mostra il programma byte dopo byte, esattamente come si trova in memoria; puoi confrontare con la pag. 267 del primo volume. Però il contenuto del buffer dopo il GET# numero 193 ci mostra il blocco di chiusura del file, abbiamo cioè perso un pezzo del file, a dimostrare che i file di programma non possono essere letti come file ASCII.

Segue il programma FILEPROG2, che ci consente di vedere il contenuto del blocco di testata di un file di programma.

```
10 REM FILEPROG2
15 REM STAMPA TESTATA FILE PROGRAMMA
20 GOSUB100:REM AZZERA BUFFER
25 OPEN4,4:REM APRE STAMPANTE
30 OPEN1,1,0,"PRGINMEM"
35 PRINT#4,"BUFFER DOPO OPEN PRGINMEM"
40 PRINT#4:GOSUB55
45 CLOSE1:CLOSE4:STOP
50 REM UTCASS
55 REM STAMPA BUFFCASS
60 REM INDIRIZZI BUFFER CASSETTA
65 I1=828:I2=1019
70 FORI=I1TOI2STEP6
75 FORK=0TO5
80 PRINT#4,PEEK(I+K);" ";
85 NEXTK
90 PRINT#4:NEXTI
95 RETURN
100 REM AZZERA BUFFCASS
105 I1=828:I2=1019
110 FORI=I1TOI2
115 POKEI,32
120 NEXTI
125 RETURN
```

Noi abbiamo operato così:

• abbiamo salvato su cassetta il programma PRGINMEM nei diversi modi possibili, cioè con SAVE senza sa, o con sa=1, sa=2 e sa=3,

• abbiamo aperto ognuno di questi file come file in lettura e stampato il contenuto del buffer usando il programma FILEPROG2,

• abbiamo constatato che cambia il primo carattere del bloccco di testata.

Segue il programma FILEPROG3.

```
10 REM FILEPROG3
15 REM STAMPA BUFFER DOPO SAVE PROGRAMMA
20 OPEN4,4:REM APRE STAMPANTE
25 PRINT#4,"BUFFER DOPO SAVE"
```

```
30 PRINT#4:GOSUB45
35 CLOSE1:CLOSE4:STOP
40 REM UTCASS
45 REM STAMPA BUFFCASS
50 REM INDIRIZZI BUFFER CASSETTA
55 I1=828:I2=828+17
60 FORI=I1TOI2STEP6
65 FORK=0TO5
70 PRINT#4,PEEK(I+K);" ";
75 NEXTK
80 PRINT#4:NEXTI
85 RETURN
```

Con questo programma abbiamo operato così:

- abbiamo caricato in memoria FILEPROG3,
- abbiamo eseguito in immediato il SAVE del programma usando i diversi valori di sa consentiti,

- dopo ogni operazione di SAVE abbiamo fatto girare il programma ottenendo in stampa i primi 18 caratteri del buffer; anche in questo caso nel blocco di chiusura del file cambia il primo carattere.

2.4 Esempio di file SEQUENZIALE

Riportiamo ora un programma che crea e gestisce un file sequenziale su cassetta.Il nome del programma è PRFL3C e il nome del file gestito è, in questo esempio, FL3C.

Abbiamo deciso di creare un archivio di indirizzi. Il record logico del file deve comprendere i seguenti 6 campi:

COGNOME
NOME
INDIRIZZO
CAP
CITTA'
TELEFONO.

I campi possono essere di lunghezza variabile e, in conseguenza, il record logico risulta di lunghezza variabile. Abbiamo deciso di tagliare ogni campo in modo che risulti al massimo di 79 caratteri. Inoltre abbiamo deciso di tenere i record in ordine alfabetico di cognome e nome, scartando in ingresso i nomi fuori ordine, con opportuna segnalazione, e non accettando cognomi e nomi uguali.

Il numero di record gestibili dipende dalla lunghezza del nastro e dalla memoria disponibile. Devi tener presente che la casa costruttrice raccomanda di non usare cassette superiori ai 30 minuti. Inoltre, nella fase di aggiornamento del file, dato che si dispone di un solo registratore collegato, deve essere caricato in memoria tutto il file, deve essere modificato e poi riscritto tutto. La memoria occupata dipende dalla lunghezza dei campi dei record.

Il programma esegue le seguenti operazioni:

1) Crea il file ex-novo

2) Lista il file sulla stampante

3) Aggiorna il file consentendo di:

- modificare record,
- cancellare record,
- aggiungere record.

Riportiamo nella figura 2.1 lo schema a grandi blocchi del programma.
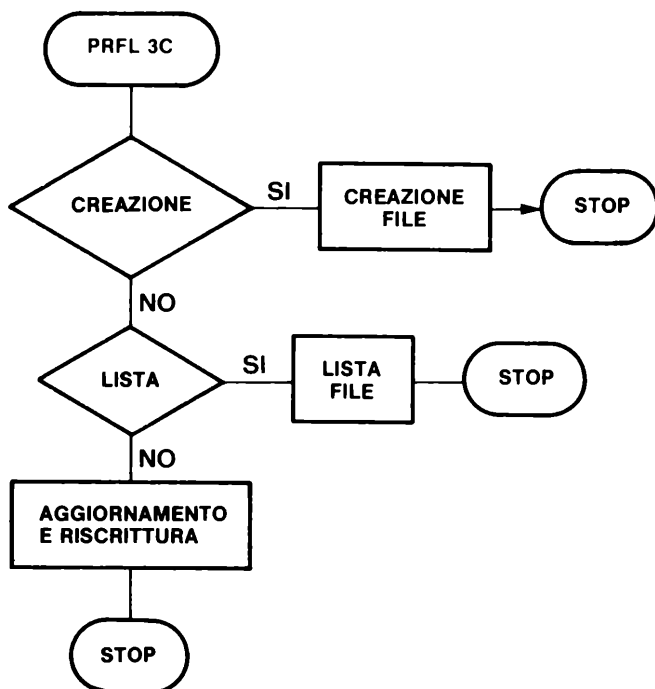
Fig. 2.1 SCHEMA A GRANDI BLOCCHI DEL PROGRAMMA

Abbiamo costruito il programma usando la tecnica dei sottoprogrammi e in modo che possa essere facilmente modificato per adattarlo alle proprie esigenze.

Il programma non presenta un menù iniziale su tutte le funzioni, ma pone domande successive sulle operazioni possibili. Finita una fase, per passare alle altre si deve ripartire con RUN. Inoltre, se, dopo la creazione del file, si vuole farne la lista o altro, si deve provvedere a riavvolgere il nastro.

Per uscire dalla fase di caricamento dati si deve rispondere con quattro asterischi alla richiesta del COGNOME (****).
Nella fase di aggiornamento viene caricato in memoria tutto il file, dimensionando opportunamente sei vettori di stringhe, dopo aver chiesto il numero dei record che compongono il file. È opportuno, dopo ogni registrazione, segnare sulla cassetta o altrove il numero dei record che sono stati memorizzati. Quando si aggiorna il file è bene conservare la cassetta con la vecchia copia e registrare la nuova su un'altra cassetta.

La fase di aggiornamento si articola dapprima nella correzione dei record esistenti, poi vengono operate eventuali cancellazioni; queste due operazioni possono essere svolte senza rispettare un ordine alfabetico, dato che la ricerca in memoria inizia sempre dalla prima posizione dei vettori. La cancellazione viene operata ponendo tre spazi al posto del cognome; in fase di riscrittura i cognomi che iniziano con spazio vengono saltati.

L'ultima fase dell'aggiornamento è l'aggiunta di nuovi record; in questo caso i nuovi record devono essere forniti in ordine alfabetico di cognome e nome, infatti il programma procede nella scrittura e aggiunge al posto giusto i nuovi record.

Riportiamo nella figura 2.2 lo schema a grandi blocchi della fase di aggiornamento.
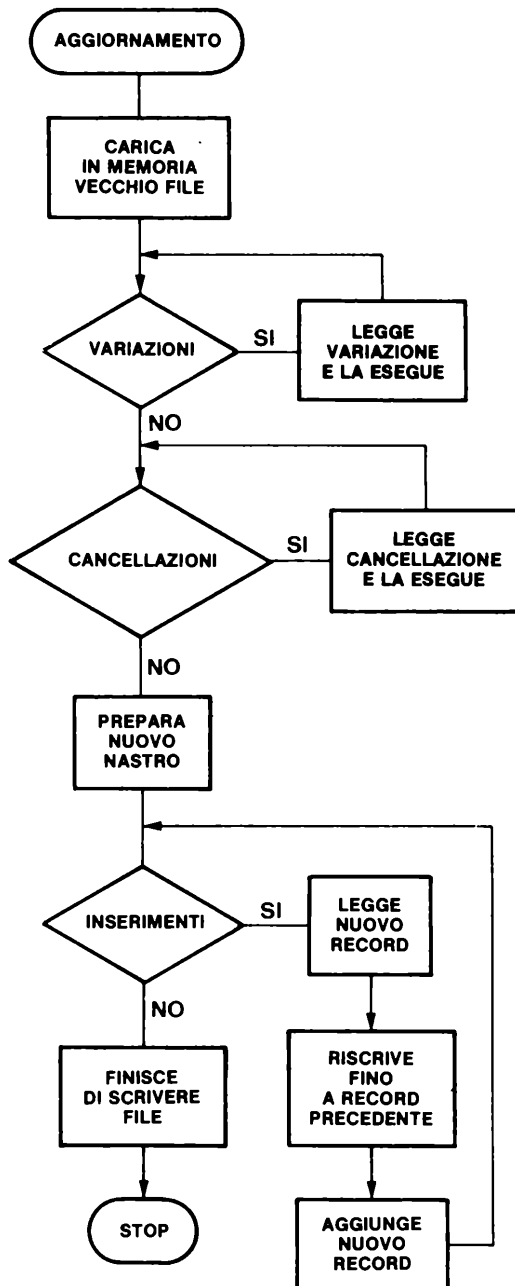
Fig. 2.2 SCHEMA A GRANDI BLOCCHI DELL'AGGIORNAMENTO

Passiamo ad elencare le variabili e le costanti usate nel programma.

PR=4, numero periferica di stampa
NF=4, lfn del file di stampa
D$(6), vettore contenente le descrizioni dei campi dati (COGNOME, NOME, INDIRIZZO, CAP, CITTA', TEL.)
I$(6), vettore contenente i campi dati di un record
CH$=CHR$(13), carattere separatore RETURN
SP$="   " costante contenente 3 caratteri spazio
R$, stringa per le risposte S/N
NF$, nome del file
N, numero massimo dei record in memoria
LC$, vecchio cognome, per controllare l'ordinamento
LN$, vecchio nome, per controllare l'ordinamento
SW, switch di controllo: 1 se finiti dati, 0 altrimenti
N1, puntatore record letti in memoria
C1$(N), vettore dei cognomi in memoria
N1$(N), vettore dei nomi in memoria
I1$(N), vettore degli indirizzi in memoria
P1$(N), vettore dei CAP in memoria
L1$(N), vettore delle CITTA' in memoria
T1$(N), vettore dei telefoni in memoria
FF, switch per gli inserimenti in coda: 1 in coda, 0 no
FS=ST, parola di stato
NR, contatore linee
K, K1, variabili di controllo.

Per passare ad un tipo diverso di record, se rimane di 6 campi, basta modificare le descrizioni alle linee 45 e 50. Se cambia il numero dei campi si devono ricercare nel programma i cicli che fanno riferimento a 6, e modificarli. Risulta più difficile modificare il criterio di ordinamento, che qui agisce sui primi due campi ed opera in senso crescente.

CORRISPONDENZA TRA I BLOCCHI E I NUMERI DI LINEA

1) Creazione del file (linee 55-120)
2) Lista ordinata del file (linee 125-185)
3) Aggiornamento del file (linee 190-400):

    3.1) Modifiche record (linee 255-275)
    3.2) Cancellazioni record (linee 280-310)
    3.3) Riscrittura e inserimenti (linee 315-400).

Segue il listato del programma PRFL3C.

```
10 REM PRFL3C
15 REM CREAZIONE E GESTIONE FILE SEQUENZIALE
20 REM NUMERO DEVICE-NUMERO LOGICO STAMPA
25 PR=4:NF=4
30 REM VETTORI DESCRIZIONI E DATI
35 DIMD$(6),I$(6)
40 CH$=CHR$(13):SP$="    "
45 D$(1)="COGNOME":D$(2)="NOME":D$(3)="INDIRIZZO"
50 D$(4)="CAP":D$(5)="CITTA'":D$(6)="TEL."
55 PRINT"■■■CREAZIONE FILE S/N":INPUT R$
60 IFR$<>"S"THEN125
65 REM***CREAZIONE FILE***
70 GOSUB580:GOSUB595:GOSUB605:GOSUB615
75 LC$="":LN$="":K1=N:FORK=1TON
80 GOSUB405
85 IFSW<>0THENK1=K:K=N:GOTO115
90 IFI$(1)>LC$THEN105
95 IFI$(1)=LC$THENIFI$(2)>LN$THEN105
100 GOSUB685:GOSUB690:GOTO80
105 LC$=I$(1):LN$=I$(2)
110 GOSUB435
115 NEXTK:CLOSE1
120 PRINT"■FINITO CARICAMENTO   ";K1;" RECORD":STOP
125 PRINT"■■■STAMPA FILE S/N":INPUTR$
130 IFR$<>"S"THEN190
135 REM***STAMPA FILE***
140 GOSUB625:GOSUB580:GOSUB595:GOSUB640:NR=4
145 PRINT#NF,"LISTA FILE ";NF$:PRINT#NF:PRINT#NF
150 GOSUB650
155 PRINT#NF,I$(1);SP$;I$(2)
160 PRINT#NF,I$(3);SP$;I$(4);SP$;I$(5);SP$;I$(6)
165 PRINT#NF:PRINT#NF:NR=NR+4
170 IFFS<>64THEN180
175 CLOSE1:CLOSENF:PRINT"■■FINITO LISTA":STOP
180 IFNR=60THENFORK=1TO6:PRINT#NF:NEXTK:NR=0
185 GOTO150
190 REM***AGGIORNAMENTO FILE***
195 PRINT"■■■AGGIORNAMENTO FILE"
200 PRINT"■■MONTA NASTRO VECCHIO"
```

```
205 GOSUB690
210 GOSUB595:GOSUB605
215 DIMC1$(N),N1$(N),I1$(N),P1$(N),L1$(N),T1$(N)
220 GOSUB640
225 FORK=1TON
230 INPUT#1,C1$(K),N1$(K)
235 INPUT#1,I1$(K),P1$(K),L1$(K),T1$(K)
240 IFST=64THENCLOSE1:K1=K:K=N:NEXTK:GOTO250
245 NEXTK:PRINT"■FILE SUPERA NUM.MASS.REC.";N:STOP
250 PRINT"■LETTI ";K1;" RECORD":SW=0:N=K1
255 PRINT"■■VARIAZIONI S/N ":INPUTR$
260 IFR$<>"S"THEN280
265 REM***MODIFICA RECORD***
270 GOSUB405:IFSW<>0THEN280
275 GOSUB445:GOTO270
280 PRINT"■CANCELLAZIONI S/N ":INPUTR$
285 IFR$<>"S"THEN320
290 REM***CANCELLAZIONE RECORD***
295 PRINT"RISPONDI **** PER      USCIRE"
300 INPUT"COGNOME ";I$(1):IFI$(1)="****"THEN320
305 INPUT"NOME ";I$(2):GOSUB475
310 GOTO300
315 REM***RISCRITTURA FILE***
320 SW=0:FF=0:K1=0:N1=1
325 PRINT"■PREPARA NUOVO NASTRO":GOSUB580
330 GOSUB595:GOSUB615
335 PRINT"■INSERIMENTI S/N ":INPUTR$
340 IFR$<>"S"THEN390
345 REM***INSERIMENTO NUOVO RECORD***
350 GOSUB405:IFSW<>0THEN385
355 IFFF=1THEN365
360 GOSUB505
365 IFI$(1)>LC$GOTO380
370 IFI$(1)=LC$ANDI$(2)>LN$THEN380
375 GOSUB685:GOTO350
380 GOSUB435:LC$=I$(1):LN$=I$(2):K1=K1+1:GOTO350
385 IFFF=1THEN395
390 GOSUB555
395 PRINT"■FINITO AGGIORNAMENTO"
400 PRINT"SCRITTI ";K1;" RECORD":CLOSE1:STOP
405 REM LETTURA NUOVI DATI
410 SW=0:GOSUB700:IFSW=1THENRETURN
```

```
415 GOSUB720
420 FORJ=1TO6:REM TAGLIA CAMPI TROPPO LUNGHI
425 IFLEN(I$(J))>79THENI$(J)=LEFT$(I$(J),79)
430 RETURN
435 REM SCRIVE NUOVO RECORD SU NASTRO
440 FORJ=1TO6:PRINT#1,I$(J);CH$;:NEXTJ:RETURN
445 REM RICERCA NOME E AGGIORNA
450 FORK=1TON:IFC1$(K)=I$(1)ANDN1$(K)=I$(2)THEN465
455 NEXTK
460 GOSUB490:RETURN
465 I1$(K)=I$(3):L1$(K)=I$(4):P1$(K)=I$(5)
470 T1$(K)=I$(6):K=N:NEXTK:RETURN
475 FORK=1TON:REM RICERCA NOME
480 IFC1$(K)=I$(1)ANDN1$(K)=I$(2)THEN500
485 NEXTK
490 PRINT"■NON TROVATO NOME ";I$(1);SP$;I$(2)
495 GOSUB690:RETURN
500 C1$(K)=SP$:K=N:NEXTK:RETURN
505 REM RICERCA POSIZIONE NUOVO RECORD
510 FORK=N1TON:IFC1$(K)=SP$THEN545
515 IFC1$(K)<I$(1)THEN540
520 IFC1$(K)=I$(1)ANDN1$(K)<I$(2)THEN540
525 IFC1$(K)=I$(1)ANDN1$(K)=I$(2)THEN535
530 N1=K:K=N:NEXTK:RETURN
535 PRINT"NOMI UGUALI":GOSUB690:GOTO530
540 GOSUB660:K1=K1+1:LC$=C1$(K):LN$=N1$(K)
545 NEXTK
550 FF=1:RETURN
555 REM TERMINA SCRITTURA FILE
560 IFN1=NANDFF=1THENRETURN
565 FORK=N1TON:IFC1$(K)=SP$THEN575
570 GOSUB660:K1=K1+1
575 NEXTK:RETURN
580 REM RICHIESTA PREPARAZIONE NASTRO
585 PRINT"■MONTA NASTRO":GOSUB690
590 RETURN
595 REM RICHIESTA NOME FILE
600 INPUT"■NOME FILE ";NF$:RETURN
605 REM RICHIESTA NUMERO RECORD
610 INPUT"■QUANTI RECORD ";N:RETURN
615 REM APERTURA FILE PER SCRIVERE
620 OPEN1,1,2,NF$:RETURN
```

```
625 REM PREPARAZIONE STAMPANTE
630 PRINT"■ACCENDI LA STAMPANTE":GOSUB690
635 OPENNF,PR:RETURN
640 REM APERTURA FILE PER LEGGERE
645 OPEN1,1,0,NF$:RETURN
650 REM LETTURA RECORD DA NASTRO
655 FORJ=1TO6:INPUT#1,I$(J):NEXTJ:FS=ST:RETURN
660 REM SCRITTURA RECORD NASTRO DA VETTORI
665 PRINT#1,C1$(K);CH$;N1$(K);CH$;I1$(K);CH$;
670 PRINT#1,P1$(K);CH$;L1$(K);CH$;T1$(K)
675 RETURN
680 REM MESSAGGIO FUORI ORDINE
685 PRINT"■FUORI ORDINE "I$(1)SP$I$(2):RETURN
690 A$="":GETA$:IFA$=""THEN690:REM ATTESA TASTO
695 RETURN
700 REM INGRESSO NUOVI DATI
705 PRINT"□";:J=1:GOSUB750
710 IFI$(1)="****"THENSW=1:RETURN
715 FORJ=2TO6:GOSUB750:NEXTJ:RETURN
720 REM CONFERMA DATI E CORREZIONE
725 PRINT"■CONFERMI S/N":INPUTR$
730 IFR$="S"THENRETURN
735 INPUT"QUALE CAMPO ";J
740 PRINTD$(J)" ";:INPUTI$(J)
745 GOSUB760:GOTO720
750 REM RICHIESTA CAMPO
755 PRINTJ;" ";D$(J);" ";:INPUTI$(J):RETURN
760 REM STAMPA DATI SUL VIDEO
765 PRINT"□";:FORJ=1TO6
770 PRINTJ;" ";D$(J);" ";I$(J):NEXTJ:RETURN
775 END
```

Dalla linea 405 alla linea 775 sono contenuti i sottoprogrammi. Quando compare un messaggio di segnalazione di una situazione, per proseguire si deve premere un tasto.

Riportiamo un pezzo di listato del file FL3C generato. La stampa è organizzata su due righe per ogni record, pensando che i singoli campi siano ragionevolmente corti, data la loro natura.

```
LISTA FILE FL3C

ARBIATI   LINO
PIAZZA DUOMO 3   22222   MILANO   78654

ALLEGRI   MARIELLA
VIALE ROSE 45   22224   CAGLIARI   678954

ARCIMBOLDI   PIERINO
PIAZZA SIENA 45   45321   MILANO   906543

ARROTINO   GIGI
VIA ROSSA 45   67543   COMO   543216

ASSOLO   MARIA
CORSO VENEZIA 1   45321   VERONA   895432

BELLINI   GIUSEPPE
PIAZZA VERDE 888   65789   BRESCIA   90213

BOLLARI   MARGHERITA
BORGO VECCHIO 90   34521   SIENA   89564
```

Riportiamo un breve commento per i sottoprogrammi.

Linee 405-430 *lettura nuovi dati

- pone SW=0 e usa il sottoprogramma in 700 per leggere il nuovo record,
- se torna con SW=1 esce perchè sono finiti i dati in ingresso,
- se no chiama il sottoprogramma in 720 per la conferma con eventuale correzione dei dati letti,
- taglia i campi troppo lunghi a 79 caratteri e esce.

Linee 435-440 *scrive nuovo record su nastro

- scrive i 6 campi separati da CHR$(13) sul nastro.

Linee 445-470 *ricerca nome e aggiorna

- ricerca cognome e nome tra i record presenti nei vettori in memoria,
- se non li trova chiama il sottoprogramma in 490 per segnalare il fatto e esce,
- se li trova sostituisce i nuovi dati a quelli vecchi e esce.

Linee 475-500 *ricerca cognome e nome

- ricerca cognome e nome in memoria,
- se li trova pone 3 spazi al posto del cognome per eliminare il record e esce,
- se non li trova stampa il messaggio di segnalazione, chiama il sottoprogramma in 690 e poi esce. Qui, linea 490, si trova un'entrata indipendente per l'ultimo pezzo del sottoprogramma.

Linee 505-550 *ricerca posizione nuovo record

- salta nella ricerca i record cancellati,
- va a riscrivere i record, che precedono per cognome e nome il nuovo record, con il sottoprogramma in 660,
- se trova cognome e nome uguali, lo segnala e esce,
- se trova la posizione esce con FF=0,
- se termina i dati memorizzati esce con FF=1.

Linee 555-575 *termina scrittura file

- se il file non è ancora terminato lo finisce di scrivere usando il sottoprogramma in 660.

Linee 580-590 *richiesta preparazione nastro

- chiede di montare il nastro e attende la pressione di un tasto per continuare.

Linee 595-600 *richiesta nome file

- richiede il nome del file da elaborare.

Linee 605-610 *richiesta numero record
- richiede il numero N dei record.

Linee 615-620 *apertura file per scrivere
- apre il file per scrivere.

Linee 625-635 *apertura stampante
- apre la stampante.

Linee 640-645 *apertura file per leggere
- apre il file per leggere.

Linee 650-655 *lettura record da nastro
- legge un record dal nastro.

Linee 660-675 *scrittura record su nastro
- scrive un record sul nastro prelevandolo dai vettori in memoria.

Linee 680-685 *messaggio dati fuori ordine
- segnala il cognome e il nome trovati fuori ordine.

Linee 690-695 *attesa tasto per continuare
- attende la pressione di un tasto per proseguire.

Linee 700-715 *lettura dati da tastiera
- chiede il primo campo (sottoprogramma in 750) e se riceve **** pone SW=1 e esce,
- se no chiede gli altri 5 campi e esce.

Linee 720-745 *conferma dati e correzione
- chiede conferma dei dati presenti sul video,
- se non riceve conferma chiede quale campo modificare e lo modifica,
- se riceve conferma esce.

Linee 750-755 *richiesta campo
- chiede un campo, lo legge e esce.

Linee 760-770 *stampa record sul video
- stampa il record sul video e esce.

CAPITOLO 3

FILE SU DISCO

3.1 Unità 1541

L'unità 1541 è una periferica intelligente; essa contiene il microprocessore 6502, che lavora in modo indipendente dopo aver ricevuto i comandi dalla CPU del calcolatore. Le parti componenti l'unità sono:

- il microprocessore 6502,
- 16K di memoria ROM, contenenti il DOS (Disk Operating System),
- 2K di memoria RAM per i buffer e le memorie di lavoro,
- l'interfaccia seriale tipo IEEE488,
- due porte di comunicazione,
- le parti elettromeccaniche necessarie.

Le due porte di comunicazione consentono di collegare più di una unità 1541 al calcolatore, si può arrivare a cinque unità disco più una stampante, che comunicano a turno tramite il bus seriale.
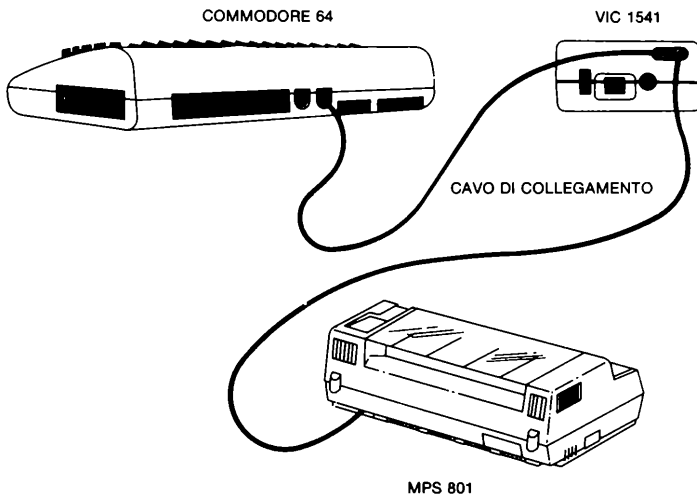
Fig. 3.1 COLLEGAMENTO TRA CALCOLATORE, UNITA' 1541 E STAMPANTE MPS801

Nella parte anteriore dell'unità sono presenti due indicatori luminosi; il più esterno, a luce verde, se acceso indica che l'unità è pronta a lavorare (READY), il più interno, a luce rossa, sta acceso mentre è in corso fisicamente una operazione di lettura o scrittura da o su floppy. Questo indicatore lampeggia quando si verifica una condizione di errore.
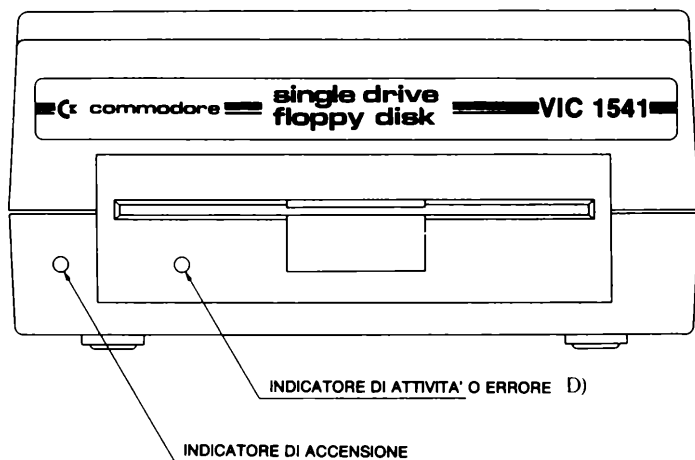
Fig. 3.2 PARTE ANTERIORE UNITA' 1541

Al COMMODORE 64 possono essere collegate fino a 5 unità 1541, in tale caso i numeri identificativi partono da 8 e possono arrivare a 12.
I dischetti (floppy) per l'unità 1541 sono gli standard 5 e 1/4", singola faccia, singola densità.

3.2 DISCHETTO

Il dischetto (floppy) è contenuto in una busta protettiva di plastica e deve essere maneggiato con cura.
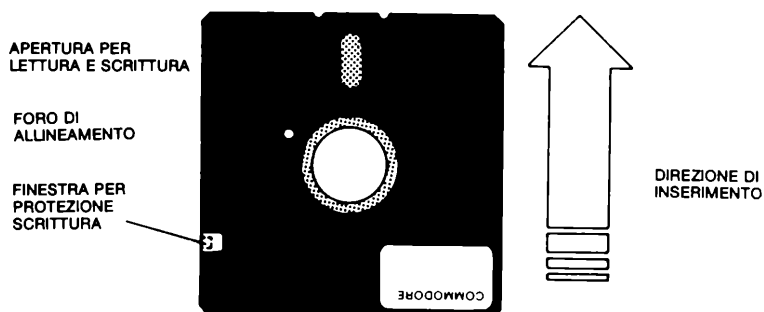
Fig. 3.3 DISCHETTO

Nella figura 3.3 sono visibili le aperture che reca la busta di plastica protettiva; la più grande serve per comunicare con la testina di lettura e scrittura, il foro piccolo serve per l'allineamento (corretta inserzione), la finestra laterale, se chiusa impedisce di scrivere sul dischetto, e, quindi, serve di protezione. Il dischetto non deve mai essere toccato nelle zone delle aperture. Esso va inserito nell'unità secondo la direzione indicata in figura, quando sulla stessa è acceso l'indicatore luminoso verde. Inoltre, il dischetto non deve mai essere estratto quando è acceso l'indicatore rosso, e non deve essere lasciato in loco quando si toglie corrente all'unità 1541.

Il dischetto viene registrato secondo tracce concentriche; sul tipo usato per l'unità 1541 sono disponibili 40 tracce su una sola faccia, ma ne vengono usate dall'unità solo 35. Su ogni traccia l'unità di informazione trasferibile in una operazione di Input o Output è il SETTORE, che è il blocco fisico di questo tipo di supporto. Il settore è una parte di traccia, delimitata da due raggi che partono dal centro.

Ovviamente le dimensioni fisiche di un settore variano tra tracce diverse, ma la quantità di informazioni registrabile è sempre la stessa, per l'utente sono disponibili 256 byte per settore. Inoltre, il numero di settori disponibili sulle diverse tracce non è sempre lo stesso.
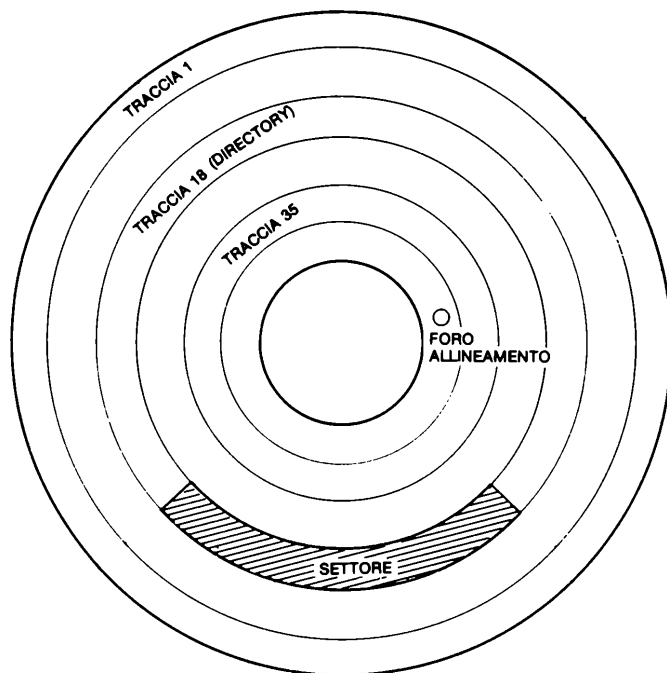
Fig. 3.4 SCHEMA NON IN SCALA DI UN DISCHETTO

Il dischetto quando è nuovo non risulta immediatamente utilizzabile; infatti su di esso non sono state registrate le suddivisioni necesssarie per poterlo usare, cioè le tracce e i settori. Il DOS fornisce il comando necessario per preparare il dischetto all'uso, tale operazione prende il nome di FORMATTAZIONE. Dopo la formattazione il dischetto reca su di sè tre tipi di indicazioni:

- quelle che permettono di distinguerlo da un altro dischetto,
- quelle che danno notizie del suo contenuto,
- quelle che marcano gli indirizzi di traccia e settore.

Il nome che identifica il dischetto è formato da due parti:

- il nome vero e proprio di 18 byte al massimo,
- la ID, identificazione del dischetto di 2 caratteri. Questa indicazione è, come vedremo, molto importante; tra l'altro consente di distinguere tra loro una serie di dischetti ai quali è stato attribuito lo stesso nome.

Nella tabella 3.1, che segue, riportiamo la suddivisione in tracce e settori.

Tab. 3.1 FORMATO DEL DISCHETTO

Numero traccia	Numero settori	Numerazione settori
da 1 a 17	21	da 0 a 20
da 18 a 24	19	da 0 a 18
da 25 a 30	18	da 0 a 17
da 31 a 35	17	da 0 a 16

Come risulta dalla tabella 3.1, sono disponibili:

17 tracce da 21 settori : 17*21 = 357
7 tracce da 19 settori : 7*19 = 133
6 tracce da 18 settori : 6*18 = 108
5 tracce da 17 settori : 5*17 = 85
----
totale settori = 683

La traccia 18, di 19 settori, ha un utilizzo particolare, che passiamo a descrivere. Il dischetto non può essere utilizzato se non contiene un indice di riferimento, come un qualunque libro, dove siano registrati i nomi dei file e i loro indirizzi sul dischetto stesso. Infatti il dischetto non è come il nastro un supporto di tipo sequenziale, su di esso si può registrare dove si vuole, e, se non si sa dove sono registrati i file, non si può utilizzarli.

La traccia 18 è utilizzata nel seguente modo:

- settore 0, byte da 0 a 143,per la mappa di occupazione del dischetto, chiamata BAM (Block Availability Map);
- settore 0, byte da 144 a 255, per l'identificazione del dischetto (blocco di testata per l'indice del disco, che viene chiamato DIRECTORY);
- settori da 1 a 19 indice del disco (DIRECTORY).

I 144 byte della BAM sono utilizzati come segue:

- byte 0 e 1, come concatenamento al settore seguente, contengono 18 e 01 (traccia 18, settore 1);
- byte 5, contiene 65, codice ASCII della lettera A, che indica il formato dell'unità, tipo 4040;
- byte 3, contiene lo 0 binario (tutti bit 0) e non ha significato,
- byte da 4 a 143, contengono la mappa dei settori occupati: sono utlizzati 4 byte per ogni traccia, quindi si hanno 35 gruppi di 4 byte (35*4=140). Ogni quartina è utilizzata così:
- primo byte: numero settori ancora disponibili nella traccia,
- secondo byte: mappa dei primi 8 settori della traccia, utilizzando i bit a partire dai meno significativi (il bit più a destra per il settore 0, il bit più a sinistra per il settore 7),
- terzo byte: mappa degli altri 8 settori della traccia dal settore 8 (bit più a destra) al settore 15 (bit più a sinistra),
- quarto byte: mappa degli altri settori della traccia a partire dal 16. Per il gruppo di tracce con 21 settori sono utilizzati gli ultimi 5 bit a destra, per il gruppo di tracce con 19 settori sono utilizzati gli ultimi 3 bit, per quello con 18 settori gli ultimi 2 e per quello con 17 settori solo l'ultimo bit.

Come vedi, la mappa del disco si serve di un bit per ogni settore, se il bit è 1, il settore è libero, se è 0, il settore è occupato.

I 112 byte della seconda parte del settore 0 sono utilizzati per l'identificazione del dischetto in questo modo:

- byte da 144 a 161, 18 byte utilizzati per il nome del disco, se il nome è più corto ad esso viene aggiunto il carattere di codice ASCII 160, corrispondente a spazio + SHIFT (spazio inverso);
- byte 161 e 162, i due caratteri della ID del disco;
- byte 163, un codice 160;
- byte 165 e 166, contengono i caratteri 2A per versione del DOS e formato dischetto;

- byte da 167 a 170, 4 codici 160;
- byte da 171 a 255, sono 85 byte non utilizzati.

I settori da 1 a 18 della traccia 18 sono utilizzati per la DIRECTORY del disco; in ogni settore è possibile registrare l'indice di 8 file. In conseguenza un disco può contenere solo 144 file diversi (18*8=144), e, se ciascuno di essi è abbastanza corto, possono restare settori inutilizzati. Come vedremo nel seguito, è anche possibile registrare settori del disco senza mantenerne indicazione nell'indice, però è una pratica non raccomandabile.

Ogni registrazione di file nell'indice prende il nome di ENTRATA (ENTRY) e occupa 30 byte.

La struttura di ogni settore è la seguente:

- byte 0 e 1, contengono il concatenamento al settore seguente, la traccia in 0 e il settore in 1; se il settore è l'ultimo della catena il byte 0 contiene lo zero binario e il byte 1 contiene il puntatore all'ultimo byte utilizzato (255 se tutto il settore),
- byte da 2 a 31, entrata 1,
- byte 32 e 33, 2 caratteri separatori contenenti zero binario,
- byte da 34 a 63, entrata 2,

....

....

- byte 224 e 225, 2 caratteri separatori,
- byte da 226 a 255, entrata 8.

Per ogni entrata di file sono registrate le seguenti informazioni:

- byte 0, tipo del file:

  tutti bit 0 per file cancellato (DEL),
  128 + 1 per sequenziale (SEQ)
  128 + 2 per programma (PRG)
  128 + 3 per utente (user) (USR)
  128 + 4 per relativo (REL),

- byte 1 e 2, indirizzo di traccia e settore del primo blocco del file,
- byte da 3 a 18, 16 byte che contengono il nome del file, se il nome è più corto sono aggiunti codici 160,
- byte 19 e 20, usati solo per i file relativi, contengono gli indirizzi di traccia e settore del primo blocco SIDE SECTOR (indice interno del file relativo),
- byte 21, usato solo per i file relativi, contiene la lunghezza del record, minore di 255,
- byte da 22 a 25, 4 byte non usati,
- byte 26 e 27, traccia e setttore del file rimemorizzato, cioè del file che è stato memorizzato con lo stesso nome dopo una OPEN@,
- byte 28 e 29, numero di blocchi occupati dal file, precede il byte basso (nel calcolo: contenuto byte 28 + 256 * contenuto byte 29).

Precedentemente abbiamo calcolato che nel dischetto sono disponibili 683 settori, dato che la traccia 18 è usata per la DIRECTORY, i settori disponibili per l'utente sono 683-19=664. Se, dopo la formattazione del dischetto, richiedi la stampa della directory, puoi vedere le indicazioni sulla identificazione del dischetto, nessuna entrata di file, e ti vengono dichiarati liberi 664 blocchi.

Quando inserisci il dischetto nell'unità 1541 vedi accendersi per poco la spia rossa e senti girare il dischetto. Avviene l'operazione di INIZIALIZZAZIONE, da non confondere con la formattazione; essa consiste nel fatto che il dischetto viene fatto girare fino ad ottenere l'allineamento della testina di lettura con l'inizio delle tracce e viene letto il settore 0 della traccia 18 in uno dei buffer di 256 byte dell'unità. Questa operazione è ESSENZIALE, infatti, se non è nota la situazione di occupazione delle tracce e dei settori, si rischia di andare a scrivere su settori già precedentemente occupati. La BAM deve risiedere nella RAM dell'unità 1541 per poter operare sul dischetto e deve essere riscritta sul dischetto quando viene modificata la situazione. Per questa ragione è assolutamente necessario chiudere i file sui quali si opera; l'operazione di chiusura fa avvenire, se necessario, la riscrittura della BAM. Un dischetto, che contenga una BAM non corrispondente alla sua reale situazione di occupazione dei settori, è praticamente inutilizzabili con le normali operazioni BASIC.

Riepiloghiamo la differenza che esiste tra le due operazioni: FORMATTAZIONE e INIZIALIZZAZIONE:

- la formattazione agisce sul supporto dischetto, annulla le precedenti registrazioni e memorizza la nuova identificazione, dopo aver registrato gli indirizzi di traccia e settore;
- l'inizializzazione allinea il dischetto nell'unità e carica nella RAM dell'unità le informazioni necessarie.

Ovviamente non viene caricata la DIRECTORY; ci vorrebbe troppo spazio. La DIRECTORY viene letta quando si apre un file; essa serve per fornire l'indirizzo di inizio del file e le altre informazioni necessarie per il tipo di file in uso. Essa viene in parte riscritta quando si chiude un file ed è necessario registrare modifiche nell'entrata ad esso relativa.

Abbiamo parlato di settori (BLOCCHI FISICI) di 256 byte. In realtà ogni settore contiene più di 256 byte, infatti in esso sono registrati anche alcuni caratteri di controllo, gli indirizzi di traccia e settore, un numero che rappresenta (CHECKSUM) la somma di tutti i bit 1 registrati nella parte di settore a disposizione dell'utente, e serve a controllare la bontà della registrazione. In sostanza in ogni settore sono presenti due parti: quella a disposizione del sistema, con un suo controllo di checksum, e quella a disposizione dell'utente di 256 byte.

Al programmatore interessa imparare a gestire i suoi 256 byte. Questi, in realtà, si riducono a 254, infatti il byte 0 e il byte 1 sono usati dal sistema per concatenare tra loro i settori di un file, ponendo in essi l'indirizzo della traccia e del settore logicamente seguente, che di solito non è quello fisicamente adiacente. Approfondiremo meglio queste cose nel seguito.
Tra i settori esiste una zona disco inutilizzata, chiamata GAP.

Riportiamo nella tabella 3.2 le caratteristiche del dischetto.

Tab. 3.2 CARATTERISTICHE TECNICHE DISCHETTO

Capacità totale: 174848 byte (683*256)
Capacità per file sequenziali: 168656 byte (664*256-664*2 (byte di concatenamento))
Capacità per file relativi: 167132 byte (664*256 - 664*2 (concatenamento) - 254*6 (side sector)), con numero massimo di record logici per file 65535
Entrate nella directory: 144
Settori per traccia: da 17 a 21
Byte per settore: 256
Tracce: 35
Totale settori: 683, di cui disponibili 664

3.3 Disk Operating System (DOS)

Il DOS (Disk Operating System) risiede nella ROM del disco; esso provvede all'esecuzione dei comandi che vengono trasmessi all'unità 1541 dal Sistema Operativo del calcolatore. Il calcolatore invia comandi o dati tramite l'interfaccia seriale, il DOS interpreta i comandi e li esegue, scrive o legge il dischetto, gestisce la BAM e la DIRECTORY.

Il linguaggio BASIC mette a disposizione i 5 comandi già noti:

OPEN, CLOSE, PRINT#, GET#, INPUT#

- il parametro "lfn", numero logico del file, ha sempre il solito significato di numero logico identificatore del file,
- il parametro "dn", se si usa una sola unità 1541, ha il valore 8,

• il parametro "sa", noto fino ad ora come "indirizzo secondario", gioca un ruolo molto importante, infatti in base al suo valore il DOS stabilisce se riceve comandi o dati dal calcolatore.

Secondo la terminologia in uso nei manuali della Commodore, il parametro "sa" viene chiamato CANALE. Nel seguito ci atteniamo anche noi a questa terminologia. Il CANALE può avere valori da 0 a 15, con i seguenti significati:

• sa=15, apre il canale 15 per invio di comandi al DOS, o per invio di segnalazioni speciali da parte del DOS;
• sa=0 e sa=1, aprono rispettivamente i canali 0 e 1 per l'esecuzione dei comandi LOAD e SAVE, cioè trasferimento di programmi tra disco e calcolatore e viceversa;
• sa=2,...,14, apre il corrispondente canale per il trasferimento di dati.

Sia con il comando OPEN, che con il comando PRINT#, possono essere inviate al DOS stringhe di comandi, come vedremo nel seguito. Quando il DOS riceve un comando che comporta l'apertura di un canale, esso assegna al canale lo spazio necessario per lavorare e mette a disposizione uno dei suoi buffer in RAM. Sull'unità sono disponibili 8 buffer di 256 byte ciascuno; di essi 4 sono impegnati per il canale dei comandi, la BAM, le variabili di lavoro e il controllo delle operazioni. Restano, in conseguenza, liberi solo 4 buffer e il numero dei file gestibili contemporaneamente risulta di 3 o di 4, a seconda del tipo di file.
Il DOS consente di gestire 4 tipi di file:

• Programmi (PRG)
• Sequenziali (SEQ)
• Random (USR, o senza specifica di tipo, cioè collegabili con gli User, che sono sequenziali, per farne comparire l'entrata nella Directory)
• Relativi (REL).

3.4 Comandi per la gestione del disco

Ci occupiamo qui dei comandi che consentono di operare con il dischetto, indipendentemente dal tipo di file coinvolto; essi possono essere usati sia in modo immediato, che da programma. Questi comandi sono già stati trattati nel volume dedicato al BASIC, ma ci sembra utile riportarli anche in questa sede per completezza nell'esposizione dell'argomento.
Si tratta di una serie di comandi da inviare al disco sul canale 15 dei comandi. L'operazione necessaria per creare la comunicazione è:

OPEN lfn,8,15

cioè abbiamo usato: dn=8 e sa=15.
Nel seguito usiamo sempre dn=8, dato che supponiamo di avere una sola unità 1541 collegata al calcolatore.
Abbiamo usato sa=15 perchè dobbiamo. aprire il canale dei comandi.
Di solito, solo per la comodità di ricordare un solo numero, noi usiamo:

OPEN 15,8,15

cioè poniamo lfn=15, ma lfn può essere un numero qualunque, purchè minore di 255, anzi meglio se minore di 127. Abitualmente per lfn si usano numeri piccoli.
A questa OPEN, che apre il canale dei comandi, può seguire una istruzione:

PRINT# lfn, stringa-comandi

che trasmette il comando al DOS. Si può, però, procedere in altro modo, incorporando la stringa dei comandi nella OPEN stessa, così:

OPEN 15,8,15,stringa-comandi

ottenendo esattamente lo stesso effetto. In ambedue i casi l'operazione deve essere conclusa con:

CLOSE 15 (o CLOSE lfn)

infatti se non adoperi la CLOSE, e in seguito usi ancora OPEN con lo stesso numero per lfn, avrai una segnalazione di errore, perchè cerchi di aprire un canale già aperto.

Dopo queste premesse, nel seguito esaminiamo le stringhe comando da trasmettere in uno dei due modi possibili. Devi tener presente che una stringa comando non deve superare 40 caratteri.

I comandi disponibili sono:

NEW, per la formattazione del dischetto
INITIALIZE, per l'inizializzazione del dischetto
VALIDATE, per sistemare la BAM in base alle entrate della Directory
COPY, per copiare un file
RENAME, per cambiare nome a un file
SCRATCH, per cancellare un file.

Ogni comando può essere citato usando il nome per esteso, o limitandosi alla prima lettera del nome.

NEW

può servire per due scopi diversi:

- preparazione di un disco nuovo per l'uso: vengono registrati gli indirizzi di traccia e settore, viene registrato il nome e la identificazione del disco, viene preparata la BAM e viene predisposto lo spazio per la Directory;
- cancellazione di un disco già usato, mantenendo la stessa identificazione, ma senza riscrivere gli indirizzi.

La stringa-comando si scrive:

"Ndr:fn,xx"

dove:

N identifica il comando e può essere anche NEW,
dr, è il numero del drive, e, se è collegato una unità con un solo disco, come la 1541, risulta 0 e può essere omesso (l'unità 4040 ha due drive, per esempio, 0 e 1),
fn, è il nome del disco,
xx, è la ID (identificazione del disco).

Per formattare un dischetto assegnandogli il nome MAGAZZINO e la ID=33 si può scrivere:

```
OPEN15,8,15,"N0:MAGAZZINO,33":CLOSE15
```

oppure:

```
OPEN15,8,15:PRINT#15,"N0:MAGAZZINO,33":CLOSE15
```

Qualora nel comando si omettano i due caratteri della ID e si usi un disco già precedentemente formattato, si ottiene la cancellazione del contenuto del dischetto, che mantiene la ID già registrata; se il nome è diverso dal precedente, esso viene modificato. Ovviamente tale operazione, diciamo ridotta, risulta più veloce di una formattazione completa. NON SI PUO' USARE IL COMANDO SENZA ID PER FORMATTARE UN DISCO NUOVO.

INIZIALIZE

serve per allineare la testina di lettura e scrittura all'inizio della traccia e per caricare la BAM nella memoria dell'unità. Meglio eseguire troppo spesso questa operazione, che non eseguirla mai. In realtà, quando si inserisce il dischetto nell'unità l'operazione di inizializzazione deve avvenire automaticamente, ma, in qualche caso, si può avere qualche intoppo. Ti raccomandiamo quindi di eseguire da

programma, o in immediato, l'inizializzazione, quando cambi il dischetto. Non eseguire da programma la funzione "I", se non hai prima chiuso i file. La stringa è molto semplice:

"I" oppure "I0" oppure "INITIALIZE"

VALIDATE

serve per rimettere ordine in un dischetto disastrato; si deve usare il comando se è successo qualcosa di irregolare, e se, per esempio, ti accorgi che la somma dei blocchi occupati più quelli liberi (basta listare la Directory) non dà 664. Una cosa irregolare può essere l'interruzione di un programma, prima che siano andate a buon fine tutte le operazioni di scrittura dei file su disco, cioè senza chiudere i file aperti (in questo caso compaiono degli asterischi nella Directory). Il commando agisce in questo modo:

- rigenera la BAM in base alle entrate regolari presenti nella Directory,
- cancella dalla Directory le entrate irregolari.

Abbiamo visto che nell'entrata di un file è registrato l'indirizzo di traccia e settore del primo blocco; i blocchi successivi sono concatenati tra loro per mezzo dei primi due byte di ogni settore, e il blocco di chiusura reca nel primo byte lo zero binario. Il sistema, seguendo la catena, può rigenerare lo stato della BAM. Da quanto abbiamo detto risulta che NON DEVE ASSOLUTAMENTE essere eseguita questa operazione se un dischetto contiene delle registrazioni, tipo file RANDOM, che non hanno una corrispondente adeguata entrata nella Directory. La stringa è molto semplice:

"V0" oppure "V" oppure "VALIDATE"

COPY

consente di ottenere copie di un file assegnandogli nomi diversi, oppure di fondere più file, fino a 4, in un unico file. Chiamiamo "dfn" il nome del file destinazione, e "sfn" il nome del file sorgente. Il file sorgente (o i file sorgenti) deve essere stato regolarmente chiuso prima di questa operazione. Le stringhe comando possibili sono:

"C0:dfn=sfn" per una copia

"C0:dfn=sfn1,sfn2,sfn3,sfn4" per fondere più file

ovviamente il file sorgente resta inalterato.

RENAME

serve per cambiare il nome di un file. L'operazione differisce dalla copiatura, infatti il file resta registrato dove si trova, ma, nella sua entrata nella Directory, viene cambiato il nome. Chiamiamo "nfn" il nome nuovo, e "ofn" il nome vecchio. Si scrive:

"R0:nfn=ofn" oppure "RENAME:nfn=ofn"

Il file ofn deve essere stato correttamente chiuso prima di operare con RENAME.

SCRATCH

cancella i file che non servono più dal disco, ponendo il tipo DEL, cioè tutti bit 0, nel primo byte della sua entrata nella Directory, e rendendo disponibili nella BAM i settori che il file occupava. Con un solo comando possono essere cancellati più file. Si scrive:

"S0:fn" per cancellare un file

"S0:fn1,fn2,...,fnl" per cancellare l file.

Ti consigliamo di usare il comando SCRATCH quando desideri riscrivere un file che esisteva precedentemente, invece di usare il carattere "@" nella OPEN, o nella SAVE, per i file di programma. Infatti il carattere "@" produce a volte risultati non desiderabili.

Per caricare in memoria la Directory del dischetto si usa questa sequenza:

OPEN15,8,15:LOAD"$",8

se vuoi listarla sul video, devi scrivere:

LIST

se, invece, vuoi listarla sulla stampante:

OPEN4,4:CMD4:LIST

e poi: PRINT#4:CLOSE4

Il sistema usa anche per i file su disco la parola di stato ST; quando ST=64, dopo una operazione di lettura, significa che si è raggiunta la segnalazione di fine file.

Il DOS consente di fare una analisi più precisa dell'andamento delle operazioni disco, richiamando tramite il canale 15 (comandi/errori) quattro variabili, che di solito si usa chiamare, con significato mnemonico: EN, EM$, ET, ES. Il significato del contenuto delle variabili è il seguente:

EN, numero del messaggio di segnalazione,
EM$, descrizione del messaggio,
ET, numero della traccia del dischetto che ha generato la segnalazione,
ES, numero del settore nella traccia.

Al posto di queste variabili, puoi usarne anche altre, esse possono essere tutte variabili stringa, oppure, come abbiamo fatto noi, solo la seconda stringa, e le altre numeriche.

Devi operare così:

```
OPEN15,8,15:PRINT#15,EN,EM$,ET,ES:CLOSE15
```

Quando scrivi un programma che gestisce file su disco devi ricordare le seguenti cose:

- Aprire il canale 15, che serve anche per la segnalazione degli errori.
- Non chiudere il canale 15, se nel programma sono ancora aperti altri canali, infatti la chiusura del canale 15 provoca la chiusura di tutti gli altri canali per il DOS, mentre nel programma (tabellina dei file aperti nella RAM del calcolatore) i file risultano ancora logicamente aperti.

- Qualora avvenga una segnalazione di errore, dal punto di vista della logica del programma, vengono chiusi tutti i file, che però restano aperti per il DOS, allora devi scrivere in immediato:
OPEN15,8,15:CLOSE15 per sistemare le cose.

- I comandi NEW, INITIALIZE e VALIDATE chiudono tutti i canali associati al disco e quindi rovinano eventuali file che non siano prima stati chiusi correttamente.

3.5 File SEQUENZIALI di dati

I file SEQUENZIALI di dati su disco hanno caratteristiche analoghe a quelle dei file sequenziali su cassetta.
L'utente deve stabilire la struttura del record logico e dei campi che lo compongono. I campi possono essere di lunghezza fissa o variabile e in conseguenza anche i record logici. L'utente non deve preoccuparsi del collegamento tra record logico e blocco fisico; esso è a carico del sistema. I dati vengono inviati dal calcolatore all'unità 1541 carattere per carattere, il DOS provvede ad ammucchiarli nel buffer messo a disposizione al momento della OPEN del file. Quando il buffer è pieno i dati vengono scritti in un settore del dischetto; i primi due byte del settore sono usati per il concatenamento al settore successivo. In questo caso non è noto a priori l'indirizzo del buffer, che risiede nella RAM dell'unità 1541.
Ogni campo deve essere separato dal campo seguente da un carattere separatore valido, che può essere la virgola (CHR$(44)) o il RETURN (CHR$(13)). Vale la regola che, se i dati devono essere letti con INPUT#, non possono essere presenti più di 79 caratteri tra due RETURN. Inoltre, se si è usato come carattere separatore registrato la virgola, tutti i campi compresi tra due RETURN devono essere letti con una sola istruzione INPUT#, che rechi nella lista un numero sufficiente di nomi di variabili. Qualora il file venga letto solo con GET#, si ha una maggiore libertà. I separatori usati nella lista dati della PRINT# tra le variabili agiscono come per la cassetta: la virgola aggiunge spazi, il punto e virgola no.

Si può interrogare la parola di stato ST per accorgersi della fine del file in lettura (ST=64), oppure si può analizzare, servendosi del canale 15, la situazione delle variabili EN, EM$, ET e ES.
I file devono essere aperti con OPEN, elaborati in scrittura o lettura, e chiusi con CLOSE. Se si opera correttamente viene registrata, nella fase di scrittura del file, una entrata nella Directory, recante tutte le indicazioni necessarie a gestire il file. Come per la cassetta, questi file possono solo essere scritti in sequenza partendo dal primo record, oppure letti in sequenza. L'aggiornamento di un file sequenziale su dischetto può solo essere ottenuto riscrivendo completamente il file. Però, nel caso del dischetto, si può lavorare meglio che con la cassetta, infatti per aggiornare un file si può:

- aprire il file preesistente in lettura,
- aprire un nuovo file con nome diverso per scrittura,
- copiare ordinatamente i record dal vecchio file nel nuovo file, apportando via via le modifiche o le aggiunte,
- chiudere alla fine i due file, cancellare il vecchio file, e cambiare il nome al nuovo file, attribuendogli il nome vecchio.

Passiamo ora ad esaminare le istruzioni BASIC disponibili.

OPEN lfn,dn,sa,"dr:fn,ft,md"

deve essere usata per aprire un file. I parametri, che possono essere costanti o variabili, hanno il seguente significato:
lfn, numero logico del file, da 1 a 127
dn, 8 per una unità 1541
sa, indirizzo secondario da 2 a 14 (canale)
dr, numero dell'unità, 0 o si omette
fn, nome del file, è quello che compare nella directory
ft, tipo=S
md, modo, può essere: W per scrivere
R per leggere.

Nella stringa, dopo sa, si può inserire prima dei ":" il carattere "@", se md=W, per ottenere di cancellare un eventuale file già presente con lo stesso nome, prima di creare il nuovo file. Ti consigliamo di usare il comando SCRATCH quando vuoi cancellare un file, e di ricorrere raramente al carattere "@".
La OPEN svolge la sua solita azione dal punto di vista del programma BASIC, e, inoltre, provoca da parte del DOS quelle azioni che sono necessarie per poter trattare un file su dischetto. Tali azioni sono diverse in dipendenza dal carattere md. Infatti, se md=R, viene cercato nella Directory il file indicato, e, se non trovato, viene segnalato un errore, mentre se md=W, e viene trovato un file con il nome indicato (in assenza del carattere @ nella stringa) viene segnalato un errore. Viene in tutti i casi predisposto un buffer per le operazioni relative al file.

PRINT#lfn,lista

provoca la scrittura dei dati della lista sul file contraddistinto dal numero logico lfn. Non ripetiamo le considerazioni già fatte sulla "lista".
Ti facciamo, però, notare, che in questo caso possono essere aperti contemporaneamente più file su disco, per lettura e scrittura, e si può operare di volta in volta su quello che interessa e viene specificato dal numero lfn. Il numero massimo di file sequenziali gestibili contemporaneamente sul dischetto è 3.

INPUT#lfn,lista

legge dal file, aperto in lettura con il numero logico lfn, tanti dati quanti corrispondono alle variabili della lista. Se i nomi delle variabili sono di tipo errato (variabili numeriche in presenza di dati non numerici), si ha segnalazione di errore. Analogamente, se i dati contengono troppi caratteri (stringhe più lunghe di 79 caratteri tra due RETURN) si ha una segnalazione di errore. Ricordiamo che si possono perdere dati, registrati separandoli con il carattere virgola, se la lista non è congrua alla situazione del file.

GET#lfn,lista

legge dal file, aperto in lettura con numero logico lfn, un carattere per volta. È opportuno che la lista contenga solo variabili stringa.

CLOSElfn

chiude il file aperto con numero logico lfn. Se il file è di scrittura, viene scritto l'ultimo blocco registrando il carattere di EOF (End Of File), e viene aggiornata la Directory. Se il file è di lettura viene chiuso senza modificare la situazione del dischetto, ma risulta ugualmente necessario chiuderlo.

Riportiamo il programma PRFSDISCO1, come esempio. Questo programma è stato ottenuto modificando il programma PRFL1C, relativo alla cassetta, per mostrare come la scrittura di un file sequenziale avviene nello stesso modo anche sul dischetto.

```
10 REM PRFSDISCO1
15 REM CREAZIONE FILE SEQUENZIALE
20 REM RECORD LOGICO FORMATO DA 3 CAMPI
25 REM CAMPI DI LUNGHEZZA VARIABILE
30 REM A$←COGNOME
35 REM B$←NOME
40 REM C$←DATO NUMERICO LETTO COME STRINGA
45 REM SEPARATORE DI CAMPO = CHR$(13)
50 REM UN SOLO PRINT SCRIVE IL RECORD
55 REM SENZA PUNTEGGIATURA FINALE
60 REM IL SISTEMA AGGIUNGE CHR$(13)
65 REM ALLA FINE DEL RECORD
70 REM ";" COME SEPARATORE TRA LE VARIABILI
75 REM PER NON AGGIUNGERE SPAZI
80 CH$=CHR$(13):OPEN2,8,2,"FSDISCO1,S,W"
85 V$=CHR$(44):PV$=CHR$(59)
90 CD$="[illegible]"
95 PRINT"♥":INPUT"COGNOME: ";A$
100 INPUT"NOME: ";B$
105 INPUT"NUMERO:";C$
110 PRINT#2,A$;CH$;B$;CH$;C$
115 PRINT#2,A$,CH$,B$,CH$,C$
120 PRINT#2,A$;V$;B$;V$;C$
125 PRINT#2,A$;PV$;B$;PV$;C$
```

```
130 R$="":PRINT CD$;"ALTRO RECORD (S/N)?";
135 GETR$:IFR$=""THEN135
140 IFR$<>"S"THEN155
145 GOTO95
155 CLOSE2
160 OPEN2,8,2,"FSDISCO1,S,R":OPEN4,4
165 PRINT#4,"FILE LETTO CON GET#":PRINT#4
170 GET#2,R$:TS=ST:PRINT#4,R$;"*";
175 IFTS<>64THEN170
180 PRINT#4:CLOSE2:CLOSE4
195 OPEN2,8,2,"FSDISCO1,S,R":OPEN4,4
200 PRINT#4,"FILE LETTO CON INPUT#":PRINT#4
205 INPUT#2,R$:TS=ST:PRINT#4,R$
210 IFTS<>64THEN205
215 PRINT#4:CLOSE2:CLOSE4:STOP
```

Abbiamo fatto girare il programma caricando due record identici a quelli usati per provare il corrispondente programma PRFL1C e abbiamo ottenuto gli stessi risultati, che seguono, senza ripetere i commenti, per i quali ti rimandiamo al Paragrafo 2.2. Ti raccomandiamo di confrontare tra loro i due programmi, per rilevare le differenze nella stesura delle operazioni relative al trattamento dei file.

```
FILE LETTO CON GET#

P*R*I*M*O*C*
*P*R*I*M*O*N*
*1*1*1*1*1*
*P*R*I*M*O*C* * * * * * * * * * *
* * * * * * * * * * *P*R*I*M*O*N* * * * * * * * * * *
* * * * * * * * * * *1*1*1*1*1*
*P*R*I*M*O*C*,*P*R*I*M*O*N*,*1*1*1*1*1*
*P*R*I*M*O*C*;*P*R*I*M*O*N*;*1*1*1*1*1*
*S*E*C*O*N*D*O*C*
*S*E*C*O*N*D*O*N*
*2*2*2*2*2*
*S*E*C*O*N*D*O*C* * * * * * * * * * *
* * * * * * * * * * *S*E*C*O*N*D*O*N* * * * * * * * * * *
* * * * * * * * * * *2*2*2*2*2*
*S*E*C*O*N*D*O*C*,*S*E*C*O*N*D*O*N*,*2*2*2*2*2*
*S*E*C*O*N*D*O*C*;*S*E*C*O*N*D*O*N*;*2*2*2*2*2*
*
```

```
FILE LETTO CON INPUT#

PRIMOC
PRIMON
11111
PRIMOC
PRIMON
11111
PRIMOC
PRIMOC;PRIMON;11111
SECONDOC
SECONDON
22222
SECONDOC
SECONDON
22222
SECONDOC
SECONDOC;SECONDON;22222
```

Analogamente a quanto fatto nel Paragrafo 2.2, riportiamo le due istruzioni 205 e 206, da usare per modificare il programma PRFSDISCO1, per non perdere campi nel record che usa, come separatore registrato, la virgola.

```
205 INPUT#2,R1$,R2$,R3$:TS=ST
206 PRINT#4,R1$:PRINT#4,R2$:PRINT#4,R3$
```

Per proporre un esempio di file sequenziale numerico, riportiamo il programma PRFSDISCO2, nel quale scriviamo come nell'esempio PRFL2C della cassetta, quattro record numerici formati da due campi ciascuno. Dopo aver chiuso il file lo apriamo in lettura, leggiamo con INPUT# e stampiamo i dati e poi, in una seconda fase, leggiamo con GET# e stampiamo i dati carattere per carattere. In questo caso non andiamo a leggere il buffer, che si trova nella RAM dell'unità 1541.

```
10 REM PRFSDISCO2
15 REM CARICAMENTO FILE NUMERICO
20 REM OGNI RECORD LOGICO COMPRENDE 2 CAMPI
25 REM IL PRIMO E' UN NUMERO INTERO
30 REM IL SECONDO UN NUMERO DECIMALE
35 REM SI USA CHR$(13) COME SEPARATORE DI CAMPO
40 REM PER USCIRE DARE UN INTERO NULLO
45 REM IL RETURN FINALE LO METTE IL SISTEMA
50 CH$=CHR$(13)
55 OPEN2,8,2,"FSDISCO2,S,W":OPEN4,4
60 PRINT"{CLR}NUMERI INTERI <= 32767{DOWN}{DOWN}"
65 PRINT"PER USCIRE RISPONDERE 0 (ZERO) A INTERO{DOWN}{DOWN}"
70 INPUT"INTERO: ";I%
75 IFI%=0THENCLOSE2:GOTO110
80 INPUT"DECIMALE: ";D
85 PRINT#2,I%;CH$;D
90 GOTO70
110 REM LETTURA FILE NUMERICO
115 REM CON INPUT VENGONO LETTI I 2 CAMPI
120 REM ESSI VENGONO STAMPATI
125 REM QUANDO E' FINITO IL FILE VIENE CHIUSO
130 REM E POI VIENE RIAPERTO, LETTO CON GET#
133 REM E STAMPATO CARATTERE PER CARATTERE
135 OPEN2,8,2,"FSDISCO2,S,R"
145 PRINT#4:PRINT#4,"CONTENUTO FILE NUMERICO"
150 PRINT#4
155 INPUT#2,I%,D:TS=ST
160 PRINT#4,I%,D
165 IFTS<>64THEN155
175 CLOSE2:PRINT#4
180 OPEN2,8,2,"FSDISCO2,S,R":I=0
183 PRINT#4,"FILE LETTO CON GET#":PRINT#4
185 GET#2,A$:TS=ST
190 PRINT#4,ASC(A$+CHR$(0));
193 I=I+1:IFI=8THENPRINT#4:I=0
195 IFTS=64 THEN CLOSE2:PRINT#4:CLOSE4:STOP
200 GOTO185
```

Riportiamo i risultati del programma.

```
CONTENUTO FILE NUMERICO

 123                 -678.43
-67                  543.78
 3456                  90.67
-43                  890.5

FILE LETTO CON GET#

 32  49  50  51  32  13  45  54
 55  56  46  52  51  32  13  45
 54  55  32  13  32  53  52  51
 46  55  56  32  13  32  51  52
 53  54  32  13  32  57  48  46
 54  55  32  13  45  52  51  32
 13  32  56  57  48  46  53  32
 13
```

Come puoi vedere, paragonando questi risultati con quelli del file su cassetta (abbiamo scritto campi identici) la registrazione avviene nello stesso modo.

Nei due esempi di programma puoi trovare applicati i comandi che abbiamo trattato in questo paragrafo. Non abbiamo fatto l'analisi degli errori tramite il canale 15, dato che si tratta solo di esempi. Ti raccomandiamo, però, di non tralasciarla in un programma elaborativo.
Osserva alla linea 190 la sequenza:
PRINT#4,ASC(A$+CHR$(0))
è necessario sommare alla stringa A$ la stringa CHR$(0) per ovviare all'inconveniente che, se A$ è la stringa nulla, la funzione ASC dà errore.

Chiudiamo questo paragrafo con il programma esempio APRISEQUENZIALI, che dimostra quanti file sequenziali possono essere aperti contemporaneamente sul dischetto.

```
1 REM APRISEQUENZIALI
5 OPEN15,8,15,"I"
7 INPUT"QUANTI FILE SEQ.";N
15 ONNGOTO60,50,40,30,20
20 OPEN6,8,6,"@0:SEQ5,S,W":GOSUB100
21 PRINT#6,"RECORDSEQ5":GOSUB100
30 OPEN5,8,5,"@0:SEQ4,S,W":GOSUB100
31 PRINT#5,"RECORDSEQ4":GOSUB100
40 OPEN4,8,4,"@0:SEQ3,S,W":GOSUB100
41 PRINT#4,"RECORDSEQ3":GOSUB100
50 OPEN3,8,3,"@0:SEQ2,S,W":GOSUB100
51 PRINT#3,"RECORDSEQ2":GOSUB100
60 OPEN2,8,2,"@0:SEQ1,S,W":GOSUB100
61 PRINT#2,"RECORDSEQ1":GOSUB100
65 ONNGOTO74,73,72,71,70
70 CLOSE6:GOSUB100
71 CLOSE5:GOSUB100
72 CLOSE4:GOSUB100
73 CLOSE3:GOSUB100
74 CLOSE2:GOSUB100
99 CLOSE15:STOP
100 INPUT#15,EN,EM$,ET,ES
110 PRINTEN;EM$;ET;ES
120 RETURN
```

Prova a far girare il programma; vedrai che dopo aver aperto 3 file ricevi la segnalazione di errore : 70 NO CHANNEL.
Il programma chiede quanti file vuoi aprire, li apre, uno dopo l'altro e poi li chiude; sul video compare il messaggio di errore o segnalazione richiesto con il sottoprogramma in 100.

Ti segnaliamo un errore del DOS, che è bene tener presente. Quando si scrive un file sequenziale, formato da record logici di 254 caratteri, compresi i fine campo, cioè con record logico coincidente con il settore, succede che dopo la chiusura del file, si perde un settore.

Abbiamo provato il programma SEQ254, che segue:

```
10 REM SEQ254
15 REM R1$ CONTIENE 70 ASTERISCHI
16 REM R2$ CONTIENE 55 ASTERISCHI
20 R1$="":FORK=1TO70:R1$=R1$+"*":NEXTK
21 R2$="":FORK=1TO55:R2$=R2$+"*":NEXTK
25 OPEN15,8,15,"I"
30 OPEN2,8,2,"PSEQ254,S,W"
35 FORK=1TO4:PRINT#2,R1$;CHR$(13);R2$;CHR$(13);
36 PRINT#2,R1$;CHR$(13);R2$;CHR$(13);
40 NEXTK
45 CLOSE2:CLOSE15
```

esso scrive 4 record formati ciascuno da 4 campi, per un totale di 254 caratteri. Dopo la chiusura del file abbiamo listato la directory del dischetto, che prima della prova è stato formattato, che riportiamo.

```
0 [illegible]
4    "PSEQ254"           SEQ
659 BLOCKS FREE.
```

Come vedi manca un blocco nel computo. Abbiamo eseguito sul dischetto la funzione VALIDATE, e abbiamo nuovamente listato la directory. Come vedi il blocco è stato ricuperato.

```
0 [illegible]
4    "PSEQ254"           SEQ
660 BLOCKS FREE.
```

Poi abbiamo provato a leggere il file sequenziale con il programma LEGGI-SEQ254, che segue.

```
10 REM LEGGISEQ254
15 REM LEGGE E STAMPA CONTENUTO FILE PSEQ254
20 K=1
25 OPEN15,8,15,"I"
30 OPEN2,8,2,"PSEQ254,S,R"
35 INPUT#2,R$:TS=ST
36 PRINT K;R$:K=K+1
40 IFTS=64THENCLOSE2:CLOSE15:STOP
45 GOTO35
```

Il file viene letto tutto senza perdita di record logici.

3.6 Esempio di archivio SEQUENZIALE

Abbiamo realizzato su disco un programma analogo a PRFL3C, mantenendone inalterate molte caratteristiche, onde consentire un confronto tra i due tipi di supporti.

Anche in questo caso abbiamo un record logico di lunghezza variabile, formato da 6 campi di lunghezza variabile. I campi sono: COGNOME, NOME, TELEFONO, INDIRIZZO, CAP e CITTA'. Non abbiamo controllato la lunghezza dei campi (tagliandoli se superano singolarmente i 79 caratteri), ma, stando alla loro natura, sembra difficile superare 79 caratteri, a meno di non farlo volutamente. In quest'ultimo caso si avrebbe segnalazione di errore in lettura.

Non si ha controllo sul numero di record caricabili; ovviamente se si scrivono moltissimi record si può superare la capacità del dischetto.
Il programma non presenta un menù iniziale, ma pone successivamente domande sull'operazione da svolgere. Le operazioni possibili sono:

1) Crea il file ex-novo
2) Lista il file sulla stampante
3) Aggiorna il file consentendo di: .inserire record,
.modificare record,
.cancellare record.

Ogni fase conclude il programma e devi ripartire con RUN per eseguire un'altra operazione. Anche le tre sottofasi della fase 3, possono essere svolte una alla volta.

Non riportiamo qui lo schema a grandi blocchi del programma, in quanto resta valido quello della Figura 2.1.

Abbiamo mantenuto la tecnica dei sottoprogrammi; in conseguenza il programma è facilmente modificabile.
Per uscire dalla richiesta dei dati, devi rispondere con "*" alla richiesta del cognome.

La più evidente differenza, rispetto all'analogo programma per la cassetta, sta nella fase dell'aggiornamento. Infatti per aggiornare il file si procede così:

- si apre un file di nome TEMP, sempre di tipo sequenziale, per scrittura, sul quale vengono via via ricopiati i record letti dal file da aggiornare,
- per inserire, devono essere forniti i nuovi record, sempre in ordine di cognome e nome, il file in lettura e quello in scrittura vengono fatti avanzare parallelamente, in modo che le inserzioni vadano al posto giusto, scartando dati forniti fuori ordine,

- per modificare record la procedura è analoga alla precedente,
- per cancellare record, il record non viene ricopiato,
- alla fine, dopo aver chiuso i due file, viene cancellato il file sorgente e viene cambiato nome al file TEMP, assegnandogli il nome del file sorgente.

Riportiamo nella Figura 3.5 uno schema a grandi blocchi della fase di aggiornamento per inserzione.
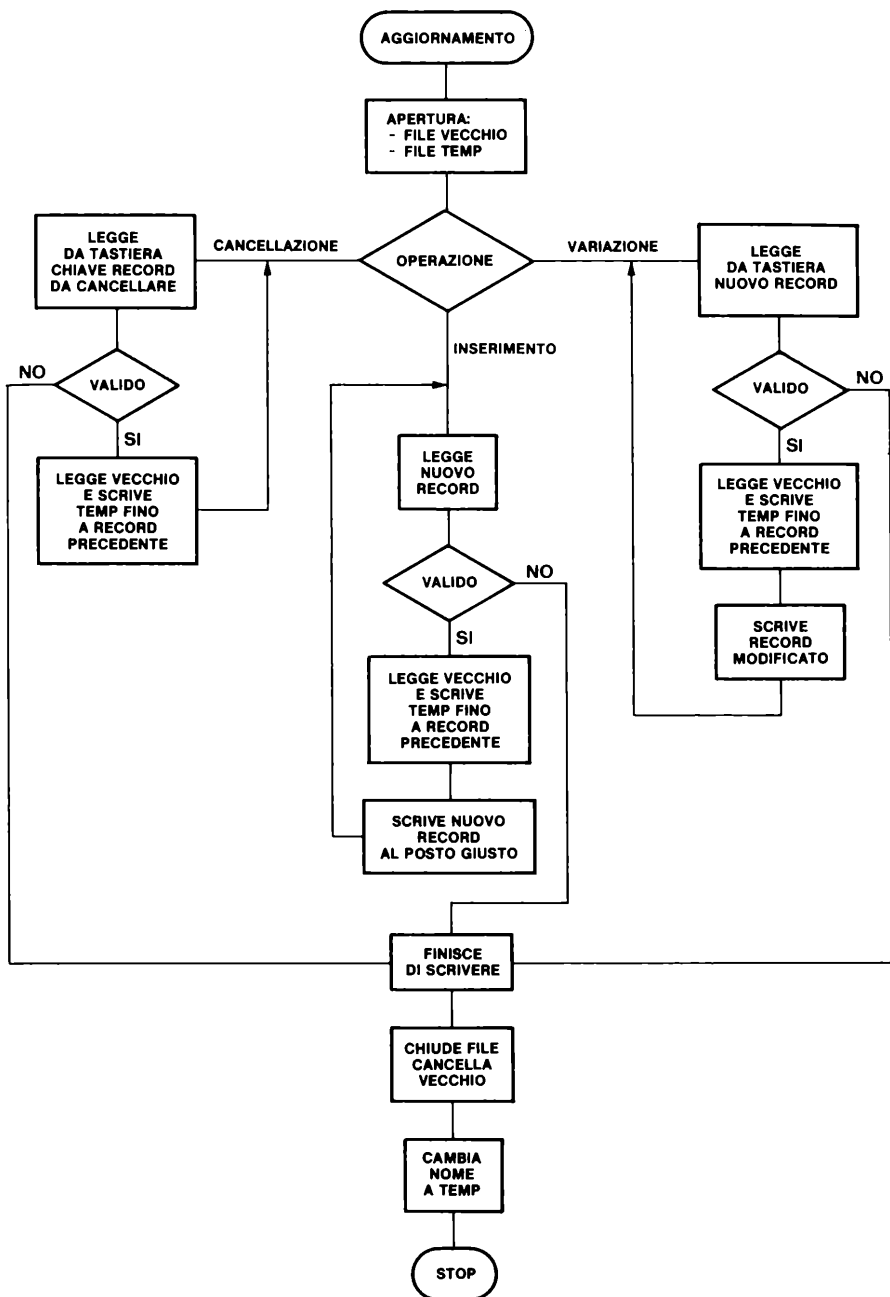
Fig. 3.5 AGGIORNAMENTO PER INSERZIONE

Ti consigliamo di dedicare un pò di tempo all'analisi di questa parte del programma; l'aggiornamento di un file sequenziale non è mai un'operazione semplice, se si vuole mantenere un ordinamento nei record.

Passiamo ad elencare le variabili e le costanti usate nel programma.

PR=4, numero periferica di stampa
NF=4, lfn del file di stampa
CH$=CHR$(13), carattere separatore RETURN
SP$="   ", 4 spazi
I$(6), vettore per leggere i dati da tastiera
I1$(6), vettore per leggere il record da disco in aggiornamento
D$(6), vettore per le descrizioni dei campi
R$, variabile per le risposte
LC$, vecchio cognome
LN$, vecchio nome
FS, memorizzazione parola di stato ST
EN, EM$, ET, ES, variabili controllo errore
SW, controllo dati ingresso, 0 per si, 1 per finiti i dati
FF, switch per fine file, 1 finito file input
FL, switch utilizzo record letto per aggiornamento, 0 utilizzato, 1 da utilizzare
R, K variabili di controllo

Segue il listato del programma.

```
10 REM PRFSDISC03
12 REM CREAZIONE E GESTIONE FILE SEQUENZIALE
14 REM PARAMETRI DI STAMPA-APERTURA CANALE 15
16 NF=4:PR=4:OPEN15,8,15:GOSUB362
18 REM COSTANTI E VARIABILI
20 CH$=CHR$(13):SP$="    "
22 DIMI$(6),I1$(6),D$(6)
24 D$(1)="COGNOME ":D$(2)="NOME ":D$(3)="TEL. "
26 D$(4)="INDIRIZZO ":D$(5)="CAP ":D$(6)="CITTA'
28 REM SCELTA OPERAZIONE
30 PRINT"CREAZIONE FILE  S/N ";:INPUTR$
32 IFR$<>"S"THEN70
34 REM------------------
36 REM CREAZIONE FILE
38 REM------------------
```

```
40 GOSUB278:GOSUB266:GOSUB282
42 PRINT#15,"I0":REM INIZIALIZZAZIONE
44 REM APERTURA FILE PER SCRIVERE-CANALE 2
46 OPEN2,8,2,"0:"+NF$+",S,W":GOSUB362
48 LC$="":LN$="":REM VECCHIO COGNOME E NOME
50 GOSUB228:REM RICHIESTA DATI
52 IFSW=0THEN60
54 CLOSE2:CLOSE15
56 PRINT"♥FINITO CARICAMENTO    FILE":STOP
58 REM CONTROLLO ORDINE RECORD
60 IFI$(1)>LC$THEN66
62 IFI$(1)=LC$THENIFI$(2)>LN$THEN66
64 GOSUB272:GOTO50
66 LC$=I$(1):LN$=I$(2)
68 GOSUB256:GOTO50
70 REM------------
72 REM STAMPA FILE
74 REM------------
76 PRINT"♥STAMPA FILE S/N";:INPUTR$
78 IFR$<>"S"THEN104
80 GOSUB278:GOSUB266
82 PRINT#15,"I0":REM INIZIALIZZAZIONE
84 GOSUB282:GOSUB286
86 OPENNF,PR:PRINT#NF,"LISTA ARCHIVIO ";NF$
88 PRINT#NF
90 REM LETTURA E STAMPA RECORD
92 GOSUB250:PRINT#NF,I$(1);SP$;I$(2)
94 PRINT#NF,I$(3):PRINT#NF,I$(4)
96 PRINT#NF,I$(5);SP$;I$(6):PRINT#NF
98 IFFS<>64THEN 90
100 CLOSE3:CLOSENF:CLOSE15
102 PRINT"♥FINITO LISTA":STOP
104 REM--------------------
106 REM AGGIORNAMENTO FILE
108 REM--------------------
110 FF=0:PRINT"♥AGGIORNAMENTO FILE"
112 GOSUB278:GOSUB266
114 PRINT#15,"I0":REM INIZIALIZZAZIONE
116 GOSUB282:GOSUB286
118 REM APERTURA FILE TEMPORANEO PER SCRIVERE
120 OPEN2,8,2,"0:TEMP,S,W":GOSUB362
122 LC$="":LN$=""
```

```
124 PRINT"]TIPO DI VARIAZIONE:"
126 PRINT"1=INSERIMENTO":PRINT"2=VARIAZIONE"
128 PRINT"3=CANCELLAZIONE"
130 INPUTR:IFR<1ORR>3THEN130
132 FL=0:REM 1 SE ULT. REC. LETTO DA SCRIVERE
134 FF=0:REM 1 SE FINITO FILE INPUT
136 ONRGOTO138,180,202:REM SCELTA OPERAZIONE
138 REM--------------------
140 REM INSERIMENTO RECORD
142 REM--------------------
144 GOSUB228:IFSW=0THEN156
146 GOSUB328:CLOSE3:CLOSE2
148 PRINT"■FINITO AGGIORNAMENTO"
150 REM SISTEMAZIONE FILE SU DISCO
152 PRINT#15,"S0:"+NF$
154 PRINT#15,"R0:"+NF$+"=TEMP":CLOSE15:STOP
156 IFI$(1)>LC$THEN170:REM INSERISCE
158 REM NOMI IN DISORDINE
160 IFI$(1)<LC$THENGOSUB272:GOTO138
162 REM COGNOMI UGUALI
164 IFI$(2)>LN$THEN170
166 IFI$(2)<LN$THENGOSUB272:GOTO138
168 GOSUB262:GOTO138
170 REM NOMI IN ORDINE
172 LC$=I$(1):LN$=I$(2):GOSUB292
174 IFI$(1)<>I1$(1)THEN178
176 IFI$(2)=I1$(2)THENGOSUB262:GOTO138
178 GOSUB256:GOTO138
180 REM--------------------
182 REM MODIFICA RECORD
184 REM--------------------
186 GOSUB228:IFSW=0THEN190
188 GOTO146
190 IFI$(1)>LC$THEN196
192 IFI$(1)=LC$THENIFI$(2)>LN$THEN196
194 GOSUB272:GOTO180
196 LC$=I$(1):LN$=I$(2):GOSUB340
198 IFFL=1THEN180
200 GOSUB256:GOTO180
202 REM---------------------------
204 REM CANCELLAZIONE RECORD
206 REM---------------------------
```

```
208 PRINTD$(1);:INPUTI$(1):IFI$(1)="*"THEN146
210 PRINTD$(2);:INPUTI$(2):IFI$(1)>LC$THEN216
212 IFI$(1)=LC$THENIFI$(2)>LN$THEN216
214 GOSUB272:GOTO202
216 LC$=I$(1):LN$=I$(2):GOSUB340
218 IFFF=1ANDFL=0THEN146
220 GOTO202
222 REM------------------
224 REM SOTTOPROGRAMMI
226 REM------------------
228 REM RICHIESTA DATI
230 SW=0:REM 1 SE FINITI DATI
232 PRINT"▯ 1 ";D$(1);:INPUTI$(1)
234 IFI$(1)="*"THENSW=1:RETURN
236 FORK=2TO6:PRINTK;D$(K);:INPUTI$(K):NEXTK
238 PRINT"▓CONFERMI S/N";:INPUTR$
240 IFR$="S"THENRETURN
242 PRINT"QUALE CAMPO ";:INPUT R
244 IFR<1ORR>6THENPRINT"▯";:GOTO242
246 PRINTD$(R);:INPUTI$(R):PRINT"▯";
248 FORK=1TO6:PRINTK;D$(K);I$(K):NEXTK:GOTO238
250 REM LETTURA RECORD DA DISCO
252 FORK=1TO6:INPUT#3,I$(K):FS=ST
254 GOSUB362:NEXTK:RETURN
256 REM SCRITTURA NUOVO RECORD
258 FORK=1TO6:PRINT#2,I$(K);CH$;
260 GOSUB362:NEXTK:RETURN
262 REM MESSAGGIO NOMI UGUALI
264 PRINT"NOMI UGUALI":GOSUB266:RETURN
266 REM ATTESA TASTO
268 GETA$:IFA$=""THEN266
270 RETURN
272 REM MESSAGGIO FUORI ORDINE
274 PRINT"▓NOMI FUORI ORDINE":GOSUB266
276 RETURN
278 REM RICHIESTA DISCO DATI
280 PRINT"▓MONTA DISCO DATI":RETURN
282 REM RICHIESTA NOME FILE
284 INPUT"NOME FILE ";NF$:RETURN
286 REM APERTURA FILE PER LEGGERE-CANALE 3
288 OPEN3,8,3,"0:"+NF$+",S,R":GOSUB362
290 RETURN
```

```
292 REM LEGGE RECORD E SCRIVE FINO AL
294 REM RECORD PRECEDENTE SU TEMP
296 IFFF=1ANDFL=0THENRETURN
298 IFFL<>0THEN304
300 GOSUB316
302 IFFS=64THENFF=1
304 IFI1$(1)<I$(1)THEN310
306 IFI1$(1)=I$(1)THENIFI1$(2)<I$(2)THEN310
308 FL=1:RETURN
310 GOSUB322:FL=0
312 IFFF=1THENRETURN
314 GOTO300
316 REM LETTURA RECORD
318 FORK=1TO6:INPUT#3,I1$(K):FS=ST
320 GOSUB362:NEXTK:RETURN
322 REM SCRITTURA RECORD
324 FORK=1TO6:PRINT#2,I1$(K)
326 GOSUB362:NEXTK:RETURN
328 REM SCRIVE FILE TEMP FINO ALLA FINE
330 REM DEL FILE DI INPUT
332 IFFF=1ANDFL=0THENRETURN
334 IFFL<>0THENFL=0:GOSUB322:GOTO328
336 GOSUB316:GOSUB322:IFFS=64THENFF=1:RETURN
338 GOTO336
340 REM COPIA FILE FINO AL RECORD CERCATO
342 IFFF=1ANDFL=0THEN354
344 IFFL<>0THEN348
346 GOSUB316:IFFS=64THENFF=1
348 IFI1$(1)<I$(1)THEN356
350 IFI1$(1)=I$(1)THENIFI1$(2)<I$(2)THEN356
352 IFI1$(1)=I$(1)ANDI1$(2)=I$(2)THENFL=0:RETURN
354 FL=1:PRINT"NON TROVATO":GOSUB266:RETURN
356 GOSUB322
358 IFFF=1THEN354
360 GOTO346
362 REM ROUTINE ERRORE
364 INPUT#15,EN,EM$,ET,ES
366 IFEN=0THEN RETURN
368 PRINT"ERRORE DISCO"
370 PRINTEN,EM$,ET,ES:STOP
```

Segue un pezzo del listato del file prova, che noi abbiamo chiamato FSDISCO3.

```
LISTA ARCHIVIO FSDISCO3

PRIMO    SIGNORE
1111111
VIA PRIMA 1
12345    MILANO

SECONDO    SIGNORE
222222
VIA SECONDA 2
65432    MILANO

TERZO    SIGNORE
333333
VIA TERZA 3
98765    MILANO
```

Nel programma non è stata particolarmente curata la lista dei record del file. I dati sono stampati su 4 linee, ponendo sulla prima il cognome e il nome, sulla seconda il telefono, sulla terza l'indirizzo e sulla quarta il CAP e la città. Non è stato fatto il controllo della lunghezza del foglio. Potrai aggiungere tu tutte queste cose, come utile esercizio.

Nel listato abbiamo messo in rilievo con REM opportune le diverse parti del programma, e, per questa ragione, non passiamo ad elencare le funzioni dei diversi gruppi di linee.

Preferiamo terminare con dei suggerimenti per migliorare il programma. La fase dell'aggiornamento potrebbbe essere impostata in modo che:

- viene chiesto un cognome e nome,
- viene chiesto se si vuole: correggere, inserire o cancellare,
- si fanno avanzare i file fino al punto giusto,
- si esegue l'operazione richiesta,
- si procede fino ad esaurire le richieste, che, ovviamente, devono essere passate in ordine di cognome e nome,
- si finisce di ricopiare il file, se necessario, e si chiude.

Con questa impostazione, le tre operazioni di aggiornamento potrebbbero essere svolte in una sola fase.

3.7 File di programmi

Abbiamo già visto nel primo volume come si trattano i file di programmi. Dopo aver studiato, nei precedenti paragrafi, il comportamento dell'unità 1541, appare evidente che un programma non può essere salvato su dischetto o richiamato da esso in memoria, se il dischetto non è stato correttamente inizializzato. Le operazioni di LOAD e SAVE usano implicitamente sa=0 e sa=1.
Devi fare attenzione a quanto segue:

- Se richiami da disco con LOAD un programma che non esiste, comincia a pulsare l'indicatore rosso di errore; in tale caso devi scrivere:

```
CLOSE15:OPEN15,8,15,"I"
```

e poi richiamare con LOAD un programma usando il nome giusto.

- Se salvi con SAVE su disco un programma con un nome che esiste già, il salvataggio non ha luogo. Puoi usare SAVE con il carattere "@", ma è meglio seguire la procedura consigliata a proposito del comando SCRATCH.

- Se non sei sicuro che una operazione disco sia andata a buon fine controlla la Directory. Ricordati che caricando in memoria la Directory si cancella il programma eventualmente presente. Questo non succede se inizialmente è stato caricato in memoria e mandato in esecuzione il programma di utilità C-64 WEDGE (vedi Paragrafo 3.15).

- Dopo il SAVE di un programma esegui sempre il VERIFY.

Nel Paragrafo 2.3 abbiamo cercato di leggere un file di programma su cassetta come file sequenziale e siamo andati ad analizzare il contenuto del buffer. Nel caso del dischetto possiamo soddisfare più facilmente la curiosità di vedere come è registrato un programma sul supporto.
Consideriamo sempre il programma PRGINMEM e operiamo come segue.
Carichiamo in memoria il programma DCOMEFS, riportato nel Paragrafo 3.15, e, dopo aver stampato la directory del dischetto che contiene il programma PRGINMEM, rileviamo l'indirizzo del primo blocco del nostro programma. Nel caso del nostro dischetto abbiamo trovato che il primo blocco sta nella traccia 19, settore 7, e che il programma occupa 2 blocchi.
Carichiamo in memoria dal dischetto TEST/DEMO, contenuto nella scatola dell'unita 1541, il programma DISPLAY T&S, montiamo il dischetto contenente il programma PRGINMEM, e facciamo girare il programma DISPLAY T&S. Chiediamo l'uscita su stampante e diamo come indirizzo di traccia e settore: 19, 7.

Otteniamo così il contenuto del primo blocco, poi continuiamo con il secondo blocco e passiamo ad analizzare lo stampato. L'interpretazione non è semplicissima, dato che non possiamo considerare la parte stampata a destra, ma solo i codici numerici riportati a sinistra in esadecimale. Se proviamo a fare qualche conversione troviamo tutti i caratteri del nostro programma, come sono riportati a pag. 267 del primo volume. I primi due byte del primo blocco contengono gli indirizzi di traccia e settore del secondo blocco, sempre in esadecimale, il secondo blocco ha l'indirizzo di traccia a zero binario per rompere la catena, mentre il secondo byte contiene il puntatore al byte di fine file nel blocco. Constatiamo così che il file di programma (tipo PRG) termina con due byte a zero binario, dopo il byte a zero binario che chiude l'ultima istruzione.

3.8 Il file con indice

Nei paragrafi e nel capitolo precedenti abbiamo trattato, illustrandoli con esempi, i file sequenziali su disco e su cassetta. Abbiamo visto che si elaborano molto semplicemente, ma che sono poco flessibili, in quanto si possono solo ricercare i record che interessano partendo dal primo e procedendo in sequenza. Nei nostri esempi di archivio, abbiamo lavorato su record organizzati secondo l'ordinamento di due campi. Ovviamente si possono anche avere archivi senza alcun ordinamento, ma in questo caso i tempi di ricerca si allungano, dato che si rischia di dover leggere tutto il file per concludere che il record cercato non esiste.

Nei prossimi paragrafi trattiamo i file RANDOM e RELATIVI, cioè file nei quali si può accedere a un qualunque record, o per mezzo del suo indirizzo fisico, o per mezzo del numero d'ordine del record nel file. Questo fatto rende la gestione molto flessibile, ma sussiste un grosso problema: come si fa a sapere qual'è l'indirizzo fisico di un record di un file RANDOM, o, analogamente, come si fa a sapere qual'è il numero d'ordine nel file RELATIVO di un record?
La prima risposta che viene in mente è la seguente: si tengono degli elenchi ordinati su carta con queste informazioni e si va a cercare quello che interessa.
La cosa non è molto elettronica!

Precisiamo alcuni concetti. Ogni file viene organizzato di solito in base a una chiave o a un gruppo di chiavi, cioè in base al contenuto di alcuni campi dei suoi record. Quando si crea il file si decide quale è la chiave di ordinamento più importante e si organizzano i record in base a tale chiave. La situazione ideale è che la chiave sia la sequenza dei numeri naturali, che risulta anche un campo significativo del record. In tale caso il campo chiave rappresenterebbe il numero d'ordine del record nel file e sarebbe immediata la ricerca su un file RELATIVO. Per un file RANDOM, non

sarebbe difficile, tenendo conto della lunghezza del record, impostare un algoritmo che consenta di risalire dal numero d'ordine del record al suo indirizzo fisico su disco, conoscendo l'indirizzo del primo blocco.

Sfortunatamente il caso citato non si presenta quasi mai. Per risolvere il problema, quando si crea un file, chiamiamolo PRINCIPALE, si crea contemporaneamente un secondo file, chiamiamolo INDICE, che ha la caratteristica di essere formato da record molto corti, contenenti il campo chiave e l'indirizzo del record corrispondente nel file principale. Il file indice, essendo corto, può essere letto in memoria con poche operazioni di lettura, e, se possibile, mantenuto in memoria durante l'esecuzione dei programmi che trattano il file principale, consentendo rapide ricerche in base alla chiave del record.
Se, per esempio, un file principale ha il record di 256 byte, mentre il campo chiave di ordinamento è di 10 byte, supponendo che per l'indirizzo bastino 3 byte, ne consegue che il file indice può avere il record di 13 byte. In un settore possono essere contenuti 16 record, quindi leggendo un settore del file indice possono essere compiute ricerche su 16 record.
Molti calcolatori, di dimensioni maggiori del nostro, hanno già incorporata nel Sistema Operativo la gestione dei file con indice. Noi possiamo crearcela su misura per le nostre esigenze.
Introdotto il concetto di file con indice, si vede immediatamente come partendo dall'indice, diciamo primario, di un file se ne può creare un altro, usando come chiave un altro campo, per ottenere una particolare elaborazione.

3.9 File RANDOM di dati

In questo contesto il file RANDOM è un file nel quale si accede ai singoli record conoscendo l'indirizzo fisico del settore al quale appartengono.
Manteniamo la solita distinzione tra record logico e record fisico. Il record logico è quello che interessa la logica del programma, ed è formato dai caratteri che servono per contenere i sui campi. Il record fisico è il settore del dischetto. Il programmatore deve aver cura di strutturare il suo record logico in modo di ADATTARLO alla struttura fisica sulla quale si deve appoggiare, per renderne il più semplice possibile la gestione.
Sarà quindi buona norma, dopo aver studiato il tracciato del record logico, aggiungere eventualmente qualche campo o qualche carattere a qualche campo già definito, in modo che la lunghezza del record logico possa:

- coincidere con il settore,
- essere un sottomultiplo del settore,
- essere un multiplo della lunghezza di un settore.

Noi, negli esempi, usiamo questa tecnica, e creiamo, insieme al file principale random, anche un file indice di tipo sequenziale.

Le elaborazioni sono più semplici se si lavora con record di lunghezza fissa, ma, dopo essersi impadroniti a fondo dell'argomento, è possibile anche lavorare con record di lunghezza variabile, ponendo come primo campo di ogni record l'informazione circa la lunghezza del record stesso.

Prendiamo ora in esame i comandi del DOS che consentono la gestione dei file RANDOM.

Per operare sui file random vengono impegnati due canali, il canale 15 dei comandi, e un altro canale (da 2 a 14) con relativo buffer. Le operazioni si svolgono in due tempi:

- SCRITTURA: le normali operazioni **PRINT#** scrivono nel buffer; una particolare operazione, lanciata sul canale comandi, trasferisce il buffer in un ben determinato settore del dischetto.
- LETTURA: una particolare operazione, lanciata sul canale comandi, trasferisce un ben determinato settore del dischetto nel buffer; le normali operazioni **INPUT#** e **GET#** leggono i dati dal buffer.

Per far partire una operazione su file sono necessarie due OPEN, una sul canale 15, e una sull'altro canale, con una stringa comando speciale.

OPEN15,8,15

apre il canale comandi, usando, secondo la nostra abitudine lfn=15.

OPENlfn,8,sa,"#"

apre il canale sa (che può avere un valore da 2 a 14), con il numero logico lfn (che può avere il valore da 1 a 127). Noi, come al solito, per semplicità, usiamo lo stesso numero per lfn e sa, da 2 a 14, escludendo, di solito, il 4, che usiamo per la stampante.
La stringa speciale "#" può essere scritta facendo seguire al carattere # un numero, il numero del buffer che si vuole utilizzare, per esempio "#3". Se richiedi un particolare buffer e questo è occupato ottieni il messaggio di errore: NO CHANNELS. Lo stesso messaggio compare se si cercano di aprire più file di quelli consentiti.
Se vuoi sapere quale buffer ti ha assegnato il sistema (avendo usato solo "#"), devi eseguire, subito dopo la OPEN, un'istruzione GET#lfn.
L'istruzione OPEN non deve precisare se si vuole leggere o scrivere; infatti in questo caso si possono fare ambedue le operazioni. La OPEN mette a disposizione un canale e un buffer; a questi si fa, come al solito, riferimento, citando il numero lfn.

Vediamo ora i comandi che si possono passare al DOS, come stringa-comando, con una operazione PRINT# sul canale 15; essi sono riportati nella Tabella 3.3.

Tab. 3.3 COMANDI PER IL DOS

COMANDI		
Completi	Abbreviati	Formato Stringa
BLOCK-READ	B-R	"B-R:"ch,dr,t,s
BLOCK-WRITE	B-W	"B-W:"ch,dr,t,s
BLOCK-EXECUTE	B-E	"B-E:"ch,dr,t,s
BUFFER-POINTER	B-P	"B-P:"ch,p
BLOCK-ALLOCATE	B-A	"B-A:"dr,t,s
BLOCK-FREE	B-F	"B-F:"dr,t,s
Memory-Write	M-W	"M-W"adl/adh/nc/data
Memory-Read	M-R	"M-R"adl/adh
Memory-Execute	M-E	"M-E"adl/adh
USER	U	"Ui:parms"

dove:

ch, numero canale, sa della OPEN... "#"
dr, numero unità: 0
t, numero traccia, da 1 a 35
s, numero settore, da 0 a 20, o meno, a seconda della traccia
p, posizione del puntatore nel buffer
adl, byte basso del blocco di memoria
adh, byte alto del blocco di memoria
nc, numero caratteri da trasferire, da 1 a 34
data, dato attuale in notazione esadecimale, si deve usare la funzione CHR$ del codice decimale equivalente
i, numero di riferimento nella tabella USER
parms, parametri associati a U

Nella stringa dei comandi devono essere usate le abbreviazioni per i comandi Memory, per gli altri si possono anche usare i nomi completi.
I parametri, se sono costanti, possono essere scritti all'interno delle virgolette, se sono variabili, no.
All'interno delle virgolette possono essere usati come caratteri separatori lo spazio e la virgola; all'esterno delle virgolette, solo il punto e virgola.

Nella Tabella 3.4 sono specificati i comandi USER.

Tab. 3.4 SALTI USER ALLA MEMORIA DELL'UNITA' 1541

TABELLA DEI SALTI		
Prima Definizione	Definizione Alternativa	Significato Comando
U1	UA	sostituisce BLOCK-READ
U2	UB	sostituisce BLOCK-WRITE
U3	UC	salto a 1280 (0500H)
U4	UD	salto a 1283 (0503H)
U5	UE	salto a 1286 (0506H)
U6	UF	salto a 1289 (0509H)
U7	UG	salto a 1292 (050CH)
U8	UH	salto a 1295 (050FH)
U9	UI	salto a 65530 (FFFAH)
U0	UJ	salto alla routine di accensione

Esaminiamo ora separatamente ogni comando, trattando prima il gruppo che si riferisce alla gestione dei file di dati, e dopo quello che interviene sulla programmazione dell'unità 1541. Non riportiamo il formato di ogni comando, rilevabile dalla Tabella 3.3.

BLOCK-READ, abbreviazione facoltativa B-R

Trasferisce il settore specificato nel buffer assegnato al momento della OPEN. Mantiene l'indicazione sulla posizione del puntatore, registrata nell'operazione di scrittura del blocco, cioè della posizione del puntatore dopo l'ultima PRINT# effettuata prima di trasferire il blocco su disco. Tale indicazione sta nella posizione

0 del buffer. Dopo il trasferimento nel buffer effettuato con B-R, se, dopo INPUT# o GET#, si analizza la parola di stato ST, quando viene raggiunta la posizione finale del puntatore si trova ST=64.

BLOCK-WRITE, abbreviazione facoltativa B-W

Trasferisce il contenuto del buffer nel settore specificato, registrando la posizione raggiunta dal puntatore con l'ultima PRINT# nella posizione 0, ma predisponendo il puntatore sulla posizione 1. Se, dopo, la lettura del blocco viene effettuata con B-R e non si sposta il puntatore, il primo carattere disponibile è quello di posizione 1, mentre se viene effettuata con U1 e non si sposta il puntatore, il primo carattere disponibile è quello di posizione zero.

BUFFER-POINTER, abbreviazione facoltativa B-P

Consente di spostare il puntatore all'interno del buffer, sia dopo avere ricevuto in esso dati in seguito a B-R o a U1, che quando si sta riempiendolo di dati prima di eseguire B-W o U2. La posizione del puntatore è molto importante, perchè le operazioni GET#, INPUT# e PRINT# agiscono a partire dalla sua posizione. Il puntatore può avere valori da 0 (primo carattere del buffer) a 255 (ultimo carattere del buffer). Se, quando usi i file random, vuoi rispettare la struttura degli altri file, non devi usare per i dati i primi due caratteri del buffer, usati negli altri casi dal sistema per concatenare tra loro i settori del disco. Quindi, prima di iniziare a scrivere, devi con B-P posizionare a 2 il puntatore, puoi, in conseguenza, usare per i dati solo 254 caratteri. Se hai operato in scrittura in questo modo, devi analogamente posizionare a 2 il puntatore prima di cominciare a leggere i dati.
Supponendo di usare un file random con il record logico di 127 caratteri, possiamo memorizzare in un settore due record logici; il primo inizia con il puntatore al valore 2, il secondo con il puntatore al valore 129.
Operando sul puntatore si può accedere con la massima libertà ai campi di un record, si deve, però fare attenzione a quale comando si usa per memorizzare il buffer su disco. Infatti, come abbiamo già visto, B-W mantiene traccia dell'ultima posizione del puntatore; vedrai che U2 non lo fa.

BLOCK-ALLOCATE, abbreviazione facoltativa B-A

registra nella BAM l'occupazione del settore specificato. Prima di poter disporre di un settore è necessario sapere se è disponibile; questo si ottiene con B-A, chiedendo di allocare un settore e analizzando dopo la situazione di errore, tramite il canale 15. Se la variabile EN contiene 65, corrispondente al messaggio NO BLOCK, significa che il settore è già occupato; ma, fortunatamente, in questo caso ET e ES contengono gli indirizzi di traccia e settore del prossimo settore disponibile. Basta usare nuovamente B-A con gli indirizzi forniti da ET e ES per allocare un settore

disponibile e renderlo indisponibile. Se EN=65 e ET=0, significa che il dischetto è completamente pieno.
Se, invece, EN contiene 0, significa che il settore richiesto è stato allocato. Riportiamo, più avanti, la routine ALLOCA che esegue correttamente questa operazione, senza andare ad usare settori, eventualmente liberi, della traccia 18.

BLOCLK-FREE, abbreviazione facoltativa B-F

libera un settore già occupato, modificando la BAM

È importante notare quanto segue:

B-A e B-F lavorano anche se il canale per l'accesso diretto al file non è aperto (cioe non è stata fatta la OPEN..."#"), ma questo è pericoloso; infatti la BAM viene riscritta quando si chiude un canale ad accesso diretto. Operando senza il canale dati aperto si rischia di non riscrivere la BAM aggiornata sul dischetto.

U1 (oppure UA)

Lavora come B-R, ma con il vantaggio che legge un intero blocco, senza tenere conto della posizione finale del puntatore nel buffer, prima dell'operazione di scrittura. Dopo U1, se non si muove il puntatore, il primo carattere disponibile è quello di posizione 0. Noi, in generale usiamo U1 per leggere, ma, in casi particolari può servire anche B-R.

U2 (oppure UB)

Lavora come B-W, ma non registra la posizione finale del puntatore nel buffer. Noi, di solito, scriviamo con U2.

Prima di passare alla descrizione dell'altro gruppo di comandi, riportiamo alcuni esempi. Se vuoi provare questi programmi devi operare così:

- prendi un dischetto nuovo, esegui l'operazione di formattazione e lo poni da parte per le prove;
- scrivi il programma esempio e lo memorizzi su un tuo dischetto per programmi, oppure carica dal dischetto dei programmi del libro il programma esempio;
- prima di dare RUN, spegni l'unità 1541, attendi un momento, poi riaccendila e monta il dischetto per le prove. Per vedere bene cosa succcede è necessario partire con la RAM dell'unità 1541 azzerata, altrimenti i buffer sono sporchi e questo può recare confusione.

Alcuni di questi programmi esempio lavorano su settori ad indirizzo fisso e si rischia di rovinare dischetti normali di lavoro.

Segue il programma RANDOMPROVE1; esso fa le seguenti cose:

- apre il canale 15, linea 40
- apre il canale 2 per i dati, linea 50
- scrive, senza intervenire sul puntatore, le tre stringhe A$, B$, C$ nel buffer, linea 60
- scrive con B-W il buffer nella traccia 20, settore 0, linea 70
- apre il canale 3 per i dati, linea 90
- scrive, senza intervenire sul puntatore, le tre stringhe A$, B$, C$ nel buffer, linea 100
- scrive con U2 il buffer nella traccia 20, settore 1
- chiude il canale 2 e il canale 3, linea 120.

Abbiamo usato due canali, aperti contemporaneamente, per ottenere di usare due buffer diversi.
Vengono scritti due settori, con gli stessi dati, usando i due comandi B-W e U2. Poi:

- trasferisce il settore 20,0 con il comando B-R nel buffer e, senza modificare il puntatore, legge i dati carattere per carattere con GET# e li stampa, linee da 125 a 150. Dai risultati vedi che la lettura con GET parte dal primo carattere del primo campo; 80 è il codice ASCII di P. Viene sentito il segnale EOF sul byte 20, l'ultimo stampato.

- trasferisce il settore 20,0 con il comando U1 nel buffer e, senza modificare il puntatore, legge i dati carattere per carattere con GET# e li stampa, linee da 155 a 180. Dai risultati vedi che la lettura con GET# parte questa volta dal carattere di posizione 0. Questo blocco è stato scritto con B-W, che registra nella posizione 0 il numero 20, puntatore all'ultima posizione scritta nel buffer (nel byte 20 trovi il 13, codice di RETURN). Leggendo con U1 non viene sentito EOF, cioè non viene preso in considerazione il contenuto del byte di posizione 0 come puntatore a fine registrazione, come vedi il buffer viene letto e stampato tutto.

- trasferisce il settore 20,1, scritto con U2, con il comando B-R nel buffer e, senza modificare il puntatore, legge i dati carattere per caratteere con GET# e li stampa, linee da 190 a 205. Dai risultati vedi che la lettura con GET# parte dal carattere di posizione 1, non sente EOF e legge tutto il buffer.

• trasferisce il settore 20,1 con il comando U1 nel buffer e, senza modificare il puntatore, legge i dati carattere per carattere con GET# e li stampa, linee da 210 a 230. Dai risultati vedi che la lettura con GET# parte dal carattere di posizione 0, che in questo caso contiene 0. Scrivendo con U2 non viene posta alcuna segnalazione in posizione 0.

Per leggere con GET#, fino alla segnalazione di EOF (se viene sentita), e stampare 8 dati per riga, si usa la routine in 300.

```
10 REM RANDOMPROVE1
20 REM DIFFERENZE TRA B-R E U1
30 REM DIFFERENZE TRA B-W E U2
33 A$="PRIMO":B$="SECONDO":C$="TERZO"
40 OPEN15,8,15,"I"
45 REM SCRIVE 20,0 CON B-W
50 OPEN2,8,2,"#"
60 PRINT#2,A$:PRINT#2,B$:PRINT#2,C$
70 PRINT#15,"B-W:2,0,20,0"
85 REM SCRIVE 20,1 CON U2
90 OPEN3,8,3,"#"
100 PRINT#3,A$:PRINT#3,B$:PRINT#3,C$
110 PRINT#15,"U2:3,0,20,1"
120 CLOSE2:CLOSE3
123 OPEN4,4:REM APRE STAMPANTE
125 REM LEGGE 20,0 CON B-R
127 PRINT#4,"LEGGE 20,0 CON B-R"
130 OPEN2,8,2,"#"
140 PRINT#15,"B-R:2,0,20,0"
150 GOSUB300:CLOSE2
155 REM LEGGE 20,0 CON U1
157 PRINT#4,"LEGGE 20,0 CON U1"
160 OPEN2,8,2,"#"
170 PRINT#15,"U1:2,0,20,0"
180 GOSUB300:CLOSE2
190 REM LEGGE 20,1 CON B-R
192 PRINT#4,"LEGGE 20,1 CON B-R"
195 OPEN2,8,2,"#"
200 PRINT#15,"B-R:2,0,20,1"
205 GOSUB300:CLOSE2
210 REM LEGGE 20,1 CON U1
```

```
212 PRINT#4,"LEGGE 20,1 CON U1"
215 OPEN2,8,2,"#"
220 PRINT#15,"U1:2,0,20,1"
230 GOSUB300:CLOSE2
250 PRINT#4:CLOSE4:CLOSE15:STOP
300 I=0
305 GET#2,D$:TS=ST:PRINT#4,ASC(D$+CHR$(0));
306 I=I+1:IFI=8THENPRINT#4:I=0
310 IFTS=64THEN PRINT#4:PRINT#4:PRINT#4:RETURN
320 GOTO 305
```

Seguono i risultati, tagliando quelli per i quali viene letto tutto il blocco, dato che, dopo i primi dati significativi, contiene tutti zeri.

```
LEGGE 20,0 CON B-R
 80  82  73  77  79  13  83  69
 67  79  78  68  79  13  84  69
 82  90  79  13


LEGGE 20,0 CON U1
 20  80  82  73  77  79  13  83
 69  67  79  78  68  79  13  84
 69  82  90  79  13  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0
 0  0
```

```
LEGGE 20,1 CON B-R
 80  82  73  77  79  13  83  69
 67  79  78  68  79  13  84  69
 82  90  79  13  0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0
```

```
LEGGE 20,1 CON U1
 0   80  82  73  77  79  13  83
 69  67  79  78  68  79  13  84
 69  82  90  79  13  0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0   0   0   0   0
 0   0   0   0
```

Possiamo concludere che:

- scrivendo con B-W viene registrato sul byte di posizione 0 il puntatore all'ultimo carattere scritto nel buffer,
- scrivendo con U2 il byte di posizione 0 non viene toccato,
- se non si danno comandi per spostare il puntatore, la scrittura dei dati inizia dal byte di posizione 1,
- la lettura con U1 non sente la segnalazione di EOF, e, senza spostare il puntatore, inizia dalla posizione 0,
- la lettura con B-R sente la segnalazione di EOF (ultima posizione raggiunta dal puntatore prima di scrivere), solo se il blocco è stato scritto con B-W, e, senza spostare il puntatore, inizia dalla posizione 1.

Se vuoi, puoi, servendoti del programma di utilità DISPLAY T&S, stampare i settori del dischetto usati nelle prove. Ricordati che il contenuto dei byte viene stampato in esadecimale.

Segue il programma RANDOMPROVE2; esso scrive sui settori 20,2 e 20,3, usando in partenza il puntatore in posizione 2, cioè saltando i primi due byte. Anche qui i due settori vengono scritti con gli stessi dati e con i due comandi disponibili, ma, la

prima volta si scrivono due volte i dati. Poi, nella fase di lettura, vengono letti, ciascuno nei due modi possibili, però, dopo il trasferimento nel buffer, viene posto il puntatore in posizione 0, ottenendo la stampa a partire dal primo byte.

```
10 REM RANDOMPROVE2
20 REM USO DEL PUNTATORE NEL BUFFER A 2
30 REM SCRITTURA CON B-W IN 20,2
31 REM SCRITTURA CON U2 IN 20,3
33 A$="PRIMO":B$="SECONDO":C$="TERZO"
40 OPEN15,8,15,"I"
45 REM SCRIVE 20,2 CON B-W
50 OPEN2,8,2,"#"
55 PRINT#15,"B-P:2,2"
60 PRINT#2,A$:PRINT#2,B$:PRINT#2,C$
65 PRINT#2,A$:PRINT#2,B$:PRINT#2,C$
70 PRINT#15,"B-W:2,0,20,2"
85 REM SCRIVE 20,3 CON U2
90 OPEN3,8,3,"#"
95 PRINT#15,"B-P:3,2"
100 PRINT#3,A$:PRINT#3,B$:PRINT#3,C$
110 PRINT#15,"U2:3,0,20,3"
120 CLOSE2:CLOSE3
123 OPEN4,4:REM APRE STAMPANTE
125 REM LEGGE 20,2 CON B-R
126 REM PUNTATORE A ZERO
127 PRINT#4,"LEGGE 20,2 CON B-R,PUNT.0"
130 OPEN2,8,2,"#"
140 PRINT#15,"B-R:2,0,20,2"
145 PRINT#15,"B-P:2,0"
150 GOSUB300:CLOSE2
155 REM LEGGE 20,2 CON U1
156 REM PUNTATORE A ZERO
157 PRINT#4,"LEGGE 20,2 CON U1,PUNT.0"
160 OPEN2,8,2,"#"
170 PRINT#15,"U1:2,0,20,2"
175 PRINT#15,"B-P:2,0"
180 GOSUB300:CLOSE2
190 REM LEGGE 20,3 CON B-R
191 REM PUNTATORE A ZERO
192 PRINT#4,"LEGGE 20,3 CON B-R,PUNT.0"
195 OPEN2,8,2,"#"
```

```
200 PRINT#15,"B-R:2,0,20,3"
201 PRINT#15,"B-P:2,0"
205 GOSUB300:CLOSE2
210 REM LEGGE 20,3 CON U1
211 REM PUNTATORE A ZERO
212 PRINT#4,"LEGGE 20,3 CON U1,PUNT.0"
215 OPEN2,8,2,"#"
220 PRINT#15,"U1:2,0,20,3"
221 PRINT#15,"B-P:2,0"
230 GOSUB300:CLOSE2
250 PRINT#4:CLOSE4:CLOSE15:STOP
300 I=0
305 GET#2,D$:TS=ST:PRINT#4,ASC(D$+CHR$(0));
306 I=I+1:IFI=8THENPRINT#4:I=0
310 IFTS=64THEN PRINT#4:PRINT#4:PRINT#4:RETURN
320 GOTO 305
```

Riportiamo i risultati, tagliando i blocchi lunghi e pieni di zeri. Abbiamo una differenza nella fase di stampa, rispetto al programma precedente, infatti, muovendo il puntatore in posizione 0 dopo il trasferimento del blocco nel buffer, la prima GET# data sul blocco, scritto con U2 e letto con B-R, interpreta lo zero iniziale come fine file e non stampa dati. Come puoi vedere il puntatore registrato in posizione 0 con il comando B-W, segna per il settore 20,2 il numero 41.

```
LEGGE 20,2 CON B-R,PUNT.0
 41  0  80  82  73  77  79  13
 83  69  67  79  78  68  79  13
 84  69  82  90  79  13  80  82
 73  77  79  13  83  69  67  79
 78  68  79  13  84  69  82  90
 79  13
```

```
LEGGE 20,2 CON U1,PUNT.0
 41  0  80  82  73  77  79  13
 83  69  67  79  78  68  79  13
 84  69  82  90  79  13  80  82
 73  77  79  13  83  69  67  79
 78  68  79  13  84  69  82  90
 79  13  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0
 0  0  0
```

```
LEGGE 20,3 CON B-R,PUNT.0
 0
```

```
LEGGE 20,3 CON U1,PUNT.0
 0  0  80  82  73  77  79  13
 83  69  67  79  78  68  79  13
 84  69  82  90  79  13  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
```

Segue il sottoprogramma ALLOCA per dimostrare l'uso corretto del comando B-A.

```
1000 REM ALLOCA
1005 REM TRACCIA TX, SETTORE SX
1010 REM SI SUPPONE APERTO UN CANALE DATI
1015 REM SI SUPPONE APERTO IL CANALE 15
1020 REM SI SUPPONE CHE SX SIA COMPATIBILE CON TX
1025 REM SCARTA LA TRACCIA 18
1030 PRINT#15,"B-A:0";TS;SX
1035 INPUT#15,EN,EM$,ET,ES
1040 IFEN=0THEN RETURN
1045 IFEN<>65THEN STOP
1050 IFET=18THENTX=19:SX=0:GOTO1030
1055 TX=ET:SX=ES:GOTO1030
```

Il sottoprogramma ALLOCA opera così:

- presuppone che sia aperto un canale dati e il canale comandi, che TX e SX contengano il numero di traccia e settore da allocare, che SX sia compatibile con il numero di settori consentiti per TX,
- cerca di allocate TX,SX, linea 1030,
- legge le quattro variabili di controllo sul canale 15, linea 1035,
- se EN=0 esce, e il settore TX,SX è stato allocato, linea 1040,
- se EN=65, significa che il settore richiesto è già occupato, allora controlla che la nuova traccia suggerita in ET non sia la 18 (directory), e, in caso cambia traccia e settore; altrimenti pone in TX e SX la traccia e il settore suggeriti in ET ed ES e va ad allocare il nuovo settore, linee 1050 e 1055,
- se si verifica un errore di altro tipo si ha uno STOP, linea 1045,
- in realtà si dovrebbe aggiungere la linea:

```
1043 IF EN=65 AND ET=0 THEN STOP
```

e questo significherebbe che viene controllato se il dischetto è pieno.

Segue il programma RANDOMPROVE3, per dimostrare l'uso di B-W e di B-R, leggendo dal byte di posizione 0 la posizione raggiunta dal puntatore, spostandolo con B-P, e continuando a registrare dati nel buffer.

```
10 REM RANDOMPROVE3
15 REM USO DELLA POSIZIONE RAGGIUNTA
20 REM DAL PUNTATORE NEL BUFFER
30 REM SCRITTURA CON B-W, LETTURA CON B-R
31 REM AGGIUNTA DATI SPOSTANDO IL PUNTATORE
32 REM IN PASE AL CONTENUTO DELLA POSIZIONE 0
33 A$="PRIMO":B$="SECONDO"
40 OPEN15,8,15,"I"
45 REM SCRIVE 20,4 CON B-W PRIMO DATO
50 OPEN2,8,2,"#"
55 PRINT#15,"B-P:2,2"
60 PRINT#2,A$
70 PRINT#15,"B-W:2,0,20,4"
80 CLOSE2
83 OPEN4,4:REM APRE STAMPANTE
84 PRINT#4,"DOPO SCRITTURA PRIMO DATO"
85 REM LEGGE 20,3 CON B-R
87 REM SPOSTANDO IL PUNTATORE IN POSIZIONE 0
```

```
90 OPEN2,8,2,"#"
93 PRINT#15,"B-R:2,0,20,4"
94 PRINT#15,"B-P:2,0":GET#2,P$
95 X=ASC(P$+CHR$(0)):X=X+1
96 PRINT#4,"PUNTATORE: ";X-1
99 GOSUB300
100 REM SPOSTA IL PUNTATORE AL VALORE LETTO IN P$
105 REM PIU' 1, PER SCRIVERE SECONDO DATO
110 PRINT#15,"B-P:2";X
115 PRINT#2,B$
120 PRINT#15,"B-W:2,0,20,4"
125 CLOSE2
130 REM LEGGE 20,3 CON B-R
135 PRINT#4,"DOPO SCRITTURA SECONDO DATO"
140 OPEN2,8,2,"#"
145 PRINT#15,"B-R:2,0,20,4"
147 PRINT#15,"B-P:2,0"
150 GOSUB300:CLOSE2
250 PRINT#4:CLOSE4:CLOSE15:STOP
300 I=0
305 GET#2,D$:TS=ST:PRINT#4,ASC(D$+CHR$(0));
306 I=I+1:IFI=8THENPRINT#4:I=0
310 IFTS=64THEN PRINT#4:PRINT#4:PRINT#4:RETURN
320 GOTO 305
```

Seguono i risultati di RANDOMPROVE3.

```
DOPO SCRITTURA PRIMO DATO
PUNTATORE:  7
 0  80  82  73  77  79  13


DOPO SCRITTURA SECONDO DATO
 15  0  80  82  73  77  79  13
 83  69  67  79  78  68  79  13
```

Come vedi, dopo aver scritto PRIMO, il puntatore indica 7, lo spostiamo alla posizione 8 e scriviamo SECONDO; alla fine il byte di posizione 0 reca 15.

Riepiloghiamo le operazioni da eseguire per scrivere un record su un file random:

- Aprire il canale 15.
- Aprire il canale per inviare i dati (OPEN..."#").
- Se si deve aggiornare o se il record logico nuovo non occupa un intero settore, leggere con B-R o U1, il blocco fisico a cui appartiene il record logico.
- Provvedere a spostare, o meno, il puntatore nel buffer.
- Scrivere con PRINT# nel buffer, usando lo stesso lfn usato nella OPEN..."#". Dopo ogni trasferimento di dato, il puntatore avanza alla posizione seguente l'ultimo carattere scritto (di solito il CHR$(13)).
- Scrivere il buffer sul dischetto con B-W o U2.
- Chiudere il canale dati e il canale comandi.

Riepiloghiamo le operazioni da fare per leggere un record da un file random:

- Aprire il canale 15.
- Aprire un canale per i dati con OPEN..."#".
- Leggere il blocco fisico a cui appartiene il record logico, con B-R o U1.
- Spostare il puntatore, se necessario.
- Leggere con INPUT# o con GET#, con lo stesso lfn usato nella OPEN del canale dei dati.
- Chiudere il canale dei dati e il canale 15.

Vediamo ora i comandi che consentono una programmazione più sofisticata; li raccomandiamo solo agli appassionati programmatori.

BLOCK-EXECUTE, abbreviazione facoltativa B-E

Consente di caricare in un buffer dell'unità 1541 il contenuto di un settore del dischetto, e, considerandolo un programma in linguaggio macchina, lo manda in esecuzione a partire dal byte di posizione 0 del buffer. Per un corretto funzionamento del comando, devono essere verificate le seguenti condizioni:

- il programma in linguaggio macchina deve avere la sua prima istruzione nel primo byte del settore,
- esso deve occupare al massimo 256 byte,
- il settore del dischetto deve essere stato preventivamente scritto con il comando U2, per non danneggiare il byte in posizione 0,
- il programma deve terminare logicamente con l'istruziine RTS (ReTurn from Subroutine).

Memory-Write, abbreviazione obbligatoria M-W

Consente di registrare, al massimo 34 byte, nella RAM dell'unità 1541. I parametri del comando sono in ordine: byte basso e byte alto dell'indirizzo di memoria da

dove iniziare a memorizzare, numero dei caratteri, byte da trasferire. Questi parametri devono essere passati con la funzione CHR$, avente per argomento il numero decimale che rappresenta il dato. Per esempio, se vogliamo scrivere all'indirizzo 1794, dobbiamo calcolare:

1794/256 dà come parte intera 7, byte alto
1794-7*256=1794-1792=2, byte basso,

in conseguenza i primi due parametri sono: CHR$(2)CHR$(7).

Se vogliamo trasferire 10 byte, il terzo parametro sarà: CHR$(10).

Dopo aver scritto i 10 byte del codice macchina dobbiamo prenderne il valore decimale e usarlo come argomento dei 10 CHR$ successivi.

Il comando si limita a memorizzare il codice a partire da un certo indirizzo. Per mandare il codice in esecuzione dobbiamo usare il comando M-E.

Memory-Read, abbreviazione obbligatoria M-R

Consente di leggere, tramite il canale comandi, il contenuto di un byte della memoria RAM o ROM dell'unità 1541 (possiamo considerarla equivalente alla funzione PEEK per il calcolatore). I parametri da passare sono il byte basso e il byte alto dell'indirizzo di memoria da cui vogliamo cominciare a leggere, passati con la funzione CHR$; così:

PRINT # 15,"M-R:"CHR$(x)CHR$(y)

dove y*256+x è l'indirizzo scelto per leggere.
Il comando pone sul canale 15 il contenuto del byte, di cui si è passato l'indirizzo; per leggere il contenuto si deve eseguire:

GET#15,C$

e in C$ si ha il dato. L'indirizzo viene incrementato automaticamente.

Segue il programma LETTURA1541, come esempio.

```
10 REM LETTURA1541
20 REM LETTURA MEMORIA
30 REM VIENE CHIESTO INDIRIZZO DI PARTENZA
40 REM VIENE CHIESTO NUMERO BYTE DA LEGGERE
50 OPEN15,8,15
60 INPUT"▒▒▒INDIRIZZO: ";I
70 INPUT"▒▒NUMERO BYTE: ";N
80 I1=INT(I/256):I2=I-I1*256
90 PRINT#15,"M-R:"CHR$(I2)CHR$(I1)
100 OPEN4,4:PRINT#4,"LETTURA DA: ";I
110 L=0:FORK=1TON
120 GET#15,C$
130 PRINT#4,ASC(C$+CHR$(0));
140 L=L+1:IFL=8THENPRINT#4:L=0
150 NEXTK:PRINT#4:CLOSE4:CLOSE15
160 STOP
```

Con esso puoi leggere N byte a partire da un indirizzo I.

Memory-Execute, abbreviazione obbligatoria M-E

Consente di mandare in esecuzione un programma in linguaggio macchina che si trovi memorizzato o nella ROM o nella RAM dell'unità 1541.

In particolare, questo comando deve essere usato per mandare in esecuzione i programmi memorizzati con il comando M-W. I parametri da passare con CHR$, sono il byte basso e il byte alto dell'indirizzo del primo byte del programma da eseguire.

Esistono, inoltre, i comandi di tipo USER; essi, esclusi i primi due, che abbiamo già esaminato, servono per operare dei salti a particolari indirizzi della memoria dell'unità1541. Gli indirizzi sono rilevabili dalla Tabella 3.4, prima riportata.
Tu puoi memorizzare a partire da questi indirizzi (quelli situati in RAM) le tue routine in linguaggio macchina, e poi mandarle in esecuzione con Ui, con la solita PRINT#15. I comandi da U3 a U8 fanno saltare all'interno di uno dei buffer dell'unità.

Con gli ultimi comandi visti si possono programmare operazioni che modificano i contenuti dei buffer, e, in conseguenza, dei file, e vengono eseguite all'interno dell'unità 1541.

3.10 Esempio di archivio RANDOM

Riportiamo un esempio di archivio RANDOM con indice. Il programma ARCHI-RANDOM fa le seguenti cose:

- crea su dischetto un archivio RANDOM con indice primario sequenziale,
- lista tutto l'archivio in ordine secondo l'indice primario (ordinato in senso crescente),
- aggiorna l'archivio:
  - aggiungendo record,
  - modificando record esistenti,
  - cancellando record esistenti,
- crea un indice secondario in base a un qualunque campo del record, escluso il primo,
- lista tutto l'archivio in ordine secondo l'indice secondario (ordinato in senso decrescente).

Il record logico è stato strutturato in 14 campi (NC=14), di tipo stringa:

1) Cognome	20 car.
2) Nome	15 "
3) Indirizzo	30 "
4) Città (compr. CAP)	25 "
5) Provincia	11 "
6) Telefono	10 "
7) Luogo nascita	20 "
8) Data nascita	8 "
9) Titolo studio	20 "
10) Occupazione attuale	20 "
11) Occupazione prec.	20 "
12) Sdtato civile	1 "
13) Nota 1	20 "
14) Nota 2	20 "

nel computo dei caratteri del record si deve aggiungere ad ogni campo un carattere di fine campo (noi usiamo il RETURN, CHR$(13)). In conseguenza il record logico è formato da:

21+16+31+26+12+11+21+9+21+21+21+2+21+21=254

caratteri. I campi sono tutti di lunghezza fissa; i primi due caratteri di ogni settore non sono utilizzati, il record logico coincide con il blocco fisico.

Noi abbiamo strutturato i campi per contenere il solito archivio Anagrafico, ma non è difficile cambiare il numero dei campi del record, le descrizioni dei campi, le lunghezze in caratteri di ogni campo; infatti tutte queste definizioni sono raggruppate all'inizio del programma e sono facilmente modificabili. Inoltre noi abbiamo considerato una chiave di ordinamento formata dai primi due campi di ogni record: Cognome e Nome. In base a questa chiave viene preparato l'indice primario; esso contiene il cognome, il nome e l'indirizzo, traccia e settore, dove si trova il record corrispondente del file principale.

Nella fase di creazione del file, viene preparato anche l'indice principale; esso viene costruito in memoria in 3 vettori opportunamente dimensionati, uno per la chiave, cognome + nome, uno per la traccia e l'altro per il settore. I record possono essere caricati anche non in ordine di chiave crescente, infatti il programma provvede a ordinare l'indice prima di scriverlo su disco. Il programma accetta anche cognomi e nomi uguali, e, in fase di ricerca chiede quale occorrenza della chiave si vuole (1, 2,...). Il file indice è di tipo sequenziale, formato da record di 3 campi: chiave, traccia e settore. L'indice principale viene caricato in memoria all'inizio dell'elaborazione (salvo che nella fase di creazione del file), può venire modificato, e viene riscritto su disco quando si termina l'elaborazione. L'indice principale, di nome INDI1, viene mantenuto in ordine di chiave crescente.

Inoltre si può costruire un indice secondario, di nome INDI2, scegliendo come campo chiave un campo qualunque del record, salvo il primo. Tale indice viene ordinato in senso decrescente e memorizzato su disco. Si può ottenere una lista dell'archivio secondo INDI2. L'indice secondario può essere rifatto tutte le volte che si vuole, cancellando il precedente.

Il programma usa un dischetto apposito solo per i dati; esso registra sulla traccia 1, settore 0, i dati generali del file, e cioè: nome dell'archivio, che coincide con il nome del disco, data di aggiornamento, numero record presenti, indirizzo di traccia e settore dell'ultimo record registrato. I record sono registrati a partire dalla traccia 1, settore 1.
Nella directory del dischetto dati sono registrati solo i file indice, INDI1 e INDI2, che sono dei normali file sequenziali. Il file random non compare nella directory; per questa ragione sul dischetto dati non si deve mai fare l'operazione VALIDATE. Infatti tale operazione cancella tutti i settori occupati, ma non registrati nella directory e ne rende disponibile lo spazio.

Il problema dell'aggiornamento del file risulta molto più semplice che nel caso del file sequenziale, dato che si può accedere a qualunque record, dopo averne trovato l'indirizzo servendosi del file indice. Ti sembrerà ridondante aver scritto il cognome e il nome sia nel record principale che nel file indice INDI1; ma abbiamo preferito avere un record completo nel file principale, cosa, del resto, necessaria per poter creare indici secondari.

Segue il listato del programma; come puoi osservare abbiamo cercato di impaginare il programma in modo che sia abbastanza facile trovare i diversi blocchi. Inoltre, il programma è stato scritto in modo modulare; questo rende possibile operare le variazioni senza troppi problemi. Ti consigliamo di adattare il programma alle tue esigenze. L'unico blocco che è molto legato alla struttura del nostro record è quello relativo alla stampa, ma è semplice da modificare.
Devi, però, ricordare che il programma si basa su un record logico coincidente con il settore, e che la somma delle lunghezze dei campi, compreso il carattere di fine campo, non deve superare 254.

```
10 REM ARCHIRANDOM
15 REM----------------------------------
20 REM DEFINIZIONE COSTANTI E VARIABILI
25 REM----------------------------------
30 NC=14:REM **NUMERO CAMPI RECORD**
35 REM          ***********************
40 S1$="DATA ULTIMO AGG.        "
45 S2$="FINITO SPAZIO ASSEGNATO"
50 SP$="                            "
55 CH$=CHR$(13):LIM$=CHR$(99)+CHR$(99)+CHR$(99)
60 DIMD$(NC):DIMY$(NC):DIML(NC)
65 DATA"COGNOME:","NOME    :","INDIR.  :"
70 DATA"CITTA'  :","PROV.   :","TELEF.  :"
75 DATA"L.NASC.:","D.NASC.:","T.STUD.:"
80 DATA"OCC.AT.:","OCC.PR.:","ST.CIV.:"
85 DATA"NOTA 1  :","NOTA 2  :"
90 DATA20,15,30,25,11,10,20,8,20,20,20,1,20,20
95 FORK=1TONC:READD$(K):NEXTK
100 FORK=1TONC:READL(K):NEXTK
105 REM **PRESENTAZIONE MENU' E SCELTE**
110 REM *******************************
115 OPEN15,8,15
120 PRINT"]QQ";TAB(10);"GESTIONE ARCHIVIO":PRINT
125 PRINTTAB(10)"1=INIZIO EX-NOVO"
130 PRINTTAB(10)"2=AGGIORNAMENTO"
135 PRINTTAB(10)"3=LISTA PRINCIPALE"
140 PRINTTAB(10)"4=CREAZ.IND.SEC."
145 PRINTTAB(10)"5=LISTA SECONDARIA"
150 PRINTTAB(10)"9=FINE"
155 INPUT"COSA SCEGLI ";X
160 IFX<1ORX>5ANDX<>9THENPRINT"]";:GOTO155
```

```
165 PRINT:GOSUB580
170 R$="":PRINTTAB(10)"MONTA DISCO DATI"
175 GETR$:IFR$=""THEN175
180 IFX=1THEN205
185 PRINT#15,"I"
190 GOSUB480:N=K:PRINT"PRESENTI ";K;" RECORD"
195 PRINTS1$;G1$;"/";M1$;"/";A1$
200 IFX<>2THEN225
205 REM-------------------------
210 REM INIZIO EX-NOVO ARCHIVIO
215 REM-------------------------
220 PRINT"      QUANTI RECORD ";:INPUT N
225 DIMC$(N),T%(N),S%(N)
230 IFX<>1THENGOSUB520
235 IFX=9THEN1290
240 ONXGOTO635,770,1095,1180,1240
245 STOP
250 REM **INGRESSO DATI**
255 REM *****************
260 PRINT"❏INGRESSO DATI"
265 FORJ=1TONC:Y$(J)="":NEXTJ
270 PRINT"PER USCIRE $ PER COGNOME";
275 PRINT"▮SE MANCANO DATIPREMI SOLO RETURN"
280 PRINTD$(1);:INPUTY$(1)
285 IFY$(1)="$"THENW=1:RETURN
290 FORJ=2TONC:PRINTD$(J);:INPUTY$(J):NEXTJ
295 REM **SISTEMA LUNGHEZZA DATI**
300 REM **************************
305 FORJ=1TONC:Y$(J)=LEFT$(Y$(J)+SP$,L(J))
310 NEXTJ:GOSUB960:RETURN
315 REM **SCRITTURA INDICE PRINCIPALE**
320 REM *******************************
325 PRINT#15,"S0:INDI1":PRINT#15,"I"
330 OPEN10,8,10,"INDI1,S,W":GOSUB550
335 FORJ=1TOK
340 PRINT#10,C$(J);CH$;T%(J);CH$;S%(J);CH$;
345 GOSUB550:NEXTJ
350 CLOSE10:RETURN
355 REM **SCRITTURA NEL BUFFER**
360 REM ************************
365 FORJ=1TONC
```

```
370 PRINT#11,Y$(J);CH$;:GOSUB550
375 NEXTJ:RETURN
380 REM **ALLOCA TRACCIA E SETTORE**
385 REM ****************************
390 PRINT#15,"B-A:"0;T;S
395 INPUT#15,EN,EM$,ET,ES
400 IFEN=0THENRETURN
405 IFEN<>65THEN570
410 IFET=18THENT=19:S=0:GOTO390
415 T=ET:S=ES:GOTO390
420 REM **PUNTATORE NEL BUFFER**
425 REM ************************
430 PRINT#15,"B-P:"11;2:GOSUB550:RETURN
435 REM **SCRITTURA RECORD**
440 REM ********************
445 PRINT#15,"U2:"11;0;T;S:GOSUB550:RETURN
450 REM **LETTURA RECORD**
455 REM ******************
460 PRINT#15,"I":OPEN11,8,11,"#":GOSUB550
465 PRINT#15,"U1:"11;0;T;S:GOSUB550:GOSUB430
470 FORJ=1TONC:INPUT#11,Y$(J):GOSUB550
475 NEXTJ:CLOSE11:RETURN
480 REM **LETT.DATI DISCO**
485 REM *******************
490 OPEN11,8,11,"#":GOSUB550
495 PRINT#15,"U1:"11;0;1;0:GOSUB550
500 GOSUB420:INPUT#11,R$,K,T,S:GOSUB550
505 G1$=LEFT$(R$,2):M1$=MID$(R$,3,2)
510 A1$=RIGHT$(R$,2):CLOSE11
515 TT=T:SS=S:RETURN
520 REM **LETT. INDICE**
525 REM ***************
530 OPEN10,8,10,"INDI1,S,R":GOSUB550
535 FORJ=1TOK
540 INPUT#10,C$(J),T%(J),S%(J):GOSUB550
545 NEXTJ:CLOSE10:RETURN
550 REM **ROUTINE ERRORE**
555 REM ******************
560 INPUT#15,EN,EM$,ET,ES
565 IFEN=0THENRETURN
570 PRINT"■ERRORE DISCO"
575 PRINTEN,EM$,ET,ES:CLOSE15:STOP
```

```
580 REM **DATA DISCO**
581 REM **************
585 PRINT"DATA PER DISCO"
590 INPUT"  GG,MM,AA█████████████";G$,M$,A$:RETURN
595 REM **SCRITT. IND. SEC.**
596 REM *********************
600 PRINT#15,"S:INDI2":PRINT#15,"I"
605 OPEN10,8,10,"INDI2,S,W"
610 FORJ=1TOK
615 PRINT#10,C$(J);CH$;T%(J);CH$;S%(J);CH$;
620 GOSUB550:NEXTJ:CLOSE10:RETURN
625 GETR$:IFR$=""THEN625
630 RETURN
635 REM------------------------
640 REM INIZIALIZZAZIONE DISCO
645 REM------------------------
650 PRINT"NOME DISCO";:INPUTN$:T=1:S=0
655 REM **DATI DISCO SETTORE 1,0**
660 REM *************************
665 PRINT#15,"N0:"+N$+",99"
670 CLOSE15:OPEN15,8,15:PRINT#15,"I"
675 REM **DATA AGG. E NUM. RECORD**
680 REM **************************
685 OPEN11,8,11,"#":GOSUB550:GOSUB390:GOSUB430
690 K=0
695 PRINT#11,G$M$A$CH$;K;CH$;1;CH$;1;CH$;
700 REM **PRIMO SETTORE DATI 1,1**
705 REM *************************
710 GOSUB550
715 PRINT#15,"U2:"11;0;T;S:GOSUB550
720 K=1:W=0:T=1:S=1
725 GOSUB250
730 IFW=1THENK=K-1:CLOSE11:GOTO1290
735 GOSUB380:GOSUB420:GOSUB355
740 REM **AGGIORNAMENTO INDICE**
745 REM ***********************
750 C$(K)=Y$(1)+Y$(2):T%(K)=T:S%(K)=S
755 GOSUB435:K=K+1
760 IFK>NTHENK=K-1:PRINTS2$:CLOSE11:GOTO1290
765 GOTO0725
770 REM-----------------
775 REM AGGIORNAMENTO
```

```
780 REM----------------
785 PRINTTAB(10);"{CLR}{DOWN}{DOWN}AGGIORNAMENTO ARCHIVIO"
790 PRINT:PRINTTAB(10);"1=CORREZIONE"
795 PRINTTAB(10);"2=AGGIUNTA ELEM."
800 PRINTTAB(10);"3=CANCELL.ELEM."
805 PRINTTAB(10);"9=FINE"
810 INPUT"COSA SCEGLI ";X:IFX=9THEN1290
815 IFX<1ORX>3THEN785
820 IFX=2THEN1010
825 IFX=3THEN1050
830 GOTO915
835 REM **RICERCA RECORD**
840 REM ******************
845 PRINT"{CLR}";D$(1);:INPUTY$(1):W=0
850 IFY$(1)="*"THENW=1:RETURN
855 PRINTD$(2);:INPUTY$(2)
860 Y$(1)=LEFT$(Y$(1)+SP$,L(1))
865 Y$(2)=LEFT$(Y$(2)+SP$,L(2))
870 INPUT"QUALE OCCORRENZA ";X
875 IFX<=0THENPRINT"{UP}";:GOTO870
880 FORJ=1TOK:IFC$(J)=Y$(1)+Y$(2)THEN900
885 NEXTJ
890 J=J-1:PRINT"NON TROVATO ";Y$(1);"  ";Y$(2)
895 GOSUB625:GOTO835
900 IFX<>1THENX=X-1:GOTO885
905 T=T%(J):S=S%(J)
910 I=J:J=K:NEXTJ:RETURN
915 REM **CORREZIONE**
920 REM **************
925 GOSUB835:IFW=1THEN785
930 GOSUB450
935 REM **VA A MODIFICA DATI**
936 REM **********************
940 GOSUB960
945 PRINT#15,"I":OPEN11,8,11,"#":GOSUB420:GOSUB355
950 C$(I)=Y$(1)+Y$(2):T%(I)=T:S%(I)=S
955 GOSUB435:CLOSE11:GOTO830
960 REM **CONTROLLO DATI E MODIFICHE**
965 REM ******************************
970 PRINT"{CLR}CONTROLLO DATI E MODIFICHE"
975 FORJ=1TONC:PRINTJ;" ";D$(J);" ";Y$(J):NEXTJ
980 INPUT"{DOWN}TUTTO BENE S/N ";X$:IFX$="S"THENRETURN
```

```
985 PRINT"QUALE CAMPO (0 USCITA)";:INPUTX
990 IFX=0THENRETURN
995 IFX>NCORX<1THENPRINT"TT":GOTO980
1000 INPUTY$(X):Y$(X)=LEFT$(Y$(X)+SP$,L(X))
1005 GOTO970
1010 REM **AGGIUNTA**
1015 REM ***********
1020 OPEN11,8,11,"#":GOSUB550
1025 PRINT"]AGGIUNTA NUOVI RECORD"
1030 PRINT"PRESENTI ";K;" RECORD"
1035 PRINT"PUOI AGGIUNGERE ";N-K;" RECORD"
1040 GOSUB625
1045 T=TT:S=SS:W=0:K=K+1:GOTO725
1050 REM **CANCELLAZIONE**
1055 REM ****************
1060 M=0
1065 GOSUB835:IFW=1THEN1085
1070 X=L(1)+L(2):C$(I)=LEFT$(LIM$+SP$+SP$,X)
1075 M=M+1
1080 PRINT#15,"B-F:"0;T;S:GOSUB550:GOTO1065
1085 REM SIST. INDICE
1090 GOSUB1355:K=K-M:GOTO1315
1095 REM------------
1100 REM LISTA FILE
1105 REM------------
1110 T$="LISTA PER INDICE PRIM."
1115 PRINT"ACCENDI STAMPANTE     PREMI UN TASTO"
1120 GETR$:IFR$=""THEN1120
1125 PRINT"]";T$:OPEN4,4
1130 PRINT#4:PRINT#4:PRINT#4,T$
1135 FORJ=1TO10:PRINT#4:NEXTJ
1140 L=2:FORI=1TOK:T=T%(I):S=S%(I):GOSUB450
1145 PRINT#4,Y$(1);"   ";Y$(2)
1150 PRINT#4,Y$(3);"   ";Y$(4);"   ";Y$(5)
1155 FORM=6TONC:PRINT#4,D$(M);Y$(M):NEXTM
1160 PRINT#4:PRINT#4
1165 IFL=5THENPRINT#4:L=0
1170 L=L+1:NEXTI
1175 CLOSE4:GOTO1340
1180 REM-----------------------------
1185 REM CREAZIONE INDICE SECONDARIO
1190 REM-----------------------------
```

```
1195 PRINT"■INDICE SECONDARIO"
1200 INPUT"■QUALE CAMPO ";X$
1205 X=VAL(X$):IFX<2ORX>NCTHENGOTO1200
1210 GOSUB315
1215 FORI=1TOK:T=T%(I):S=S%(I):GOSUB450
1220 C$(I)=Y$(X):NEXTI
1225 GOSUB595:GOSUB520
1230 PRINT"FINITO IND. SEC."
1235 GOTO1340
1240 REM--------------------------------
1245 REM LISTA PER INDICE SECONDARIO
1250 REM--------------------------------
1255 PRINT"■LISTA IND. SEC."
1260 OPEN10,8,10,"INDI2,S,R":GOSUB550
1265 FORJ=1TOK:INPUT#10,C$(J),T%(J),S%(J)
1270 GOSUB550:NEXTJ
1275 CLOSE10:GOSUB1425:GOSUB595
1280 T$="LISTA IND. SEC. "
1285 GOTO1115
1290 REM **CHIUSURA**
1295 REM ************
1300 PRINT"CHIUSURA ARCHIVIO"
1305 PRINT"IND. PRINC. DA ORDIN. S/N ":INPUTR$
1310 IFR$="S"THENGOSUB1355
1315 GOSUB315:OPEN11,8,11,"#":GOSUB550
1320 PRINT#15,"B-P:"11,2
1325 PRINT#11,G$M$A$CH$;K;CH$;T;CH$;S;CH$;
1330 GOSUB550
1335 PRINT#15,"U2:"11;0;1;0:GOSUB550:CLOSE11
1340 PRINT"FINITO AGGIORNAMENTO"
1345 PRINT"SONO PRESENTI ";K;" RECORD"
1350 CLOSE15:STOP
1355 REM **ORDINAMENTO CRESCENTE**
1360 REM **************************
1365 L=K-1
1370 W=0
1375 FORJ=1TOL
1380 IFC$(J)<=C$(J+1)THEN1405
1385 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
1390 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1395 X=S%(J):S%(J)=S%(J+1):S%(J+1)=X
1400 W=1
```

```
1405 NEXTJ
1410 IFW=0THENRETURN
1415 IFL=1THENRETURN
1420 L=L-1:GOTO1370
1425 REM **ORDINAMENTO DECRESCENTE**
1430 REM ****************************
1435 L=K-1
1440 W=0
1445 FORJ=1TOL
1450 IFC$(J)>=C$(J+1)THEN1475
1455 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
1460 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1465 X=S%(J):S%(J)=S%(J+1):S%(J+1)=X
1470 W=1
1475 NEXTJ
1480 IFW=0THENRETURN
1485 IFL=1THENRETURN
1490 L=L-1:GOTO1440
```

Il programma è formato dai seguenti blocchi:

DEFINIZIONE COSTANTI E VARIABILI
linee 15-100
sono definite alcune costanti, i vettori per contenere le descrizioni dei campi del record, le lunghezze dei campi e i dati. Si usa l'istruzione READ per riempire i vettori di costanti. Volendo modificare la struttura del record logico, si deve lavorare in questo blocco.

PRESENTAZIONE MENU' E SCELTE
linee 105-200
viene presentato il menù:
1=INIZIO EX-NOVO
2=AGGIORNAMENTO
3=LISTA PRINCIPALE
4=CREAZ.IND.SEC.
5=LISTA SECONDARIA
9=FINE
e, controllata la scelta, viene chiesto di montare il disco dati; se non è stata scelta la fase di creazione, viene segnalato quanti record sono presenti nell'archivio.

INIZIO EX-NOVO ARCHIVIO
linee 205-245 e 635-765
viene chiesto il numero di record da trattare, vengono dimensionati i tre vettori

C$(N), T%(N) e S%(N), per l'indice principale. Da questo blocco si passa sempre, qualunque sia stata la scelta. Dopo le prime operazioni il programma segue strade diverse a seconda della scelta iniziale. Se la scelta è 1, si ha:

• Formattazione del disco, con memorizzazione del nome dell'archivio (che coincide con il nome del disco), della data e dell'indirizzo iniziale del record dati (traccia 1, settore 1) nel settore 0 della traccia 1. Tale settore viene sempre aggiornato, prima della chiusura, se non altro per la data, e vi viene registrato anche il numero di record presenti nell'archivio e l'indirizzo dell'ultimo settore usato.

• Viene eseguito il sottoprogramma di richiesta dati (linea 250), e, quando i dati sono terminati, viene scritto su disco il file indice INDI1, aggiornato il settore 1,0 e il programma termina.

AGGIORNAMENTO

linee 770-1090
viene presentato il menù:

1=CORREZIONE
2=AGGIUNTA ELEM.
3=CANCELL.ELEM.
9=FINE.

Dopo il controllo della scelta, il programma va a:

• Correzione, linee 915-955

servendosi dei relativi sottoprogrammi viene chiesto il record da modificare, viene modificato, viene riscritto e viene aggiornato l'indice in memoria. Il programma consente di modificare anche i due campi chiave; ovviamente, se si modifica la chiave, è necessario riordinare l'indice.

• Aggiunta, linee 1010-1045

servendosi dei relativi sottoprogrammi, vengono aggiunti record e aggiornato l'indice. Prima di chiudere deve essere riordinato l'indice.

• Cancellazione, linee 1050-1090

i record vengono cancellati ponendo nella chiave una stringa che inizia con un codice ASCII, ripetuto 3 volte, superiore al codice delle lettere dell'alfabeto. In questo modo, ordinando l'indice in modo crescente, i record cancellati vanno a finire in fondo e poi non vengono riscritti su disco. I settori che non servono più vengono liberati con la funzione B-F.

LISTA FILE
linee 1095-1175
viene listato il file secondo un tracciato che occupa 11 linee, rispettando il cambio di foglio (66 linee). Riportiamo la lista di un record.

```
CARLINI                 GIULIO
VIALE ROSE 45                     20100 MILANO            MI
TELEF.  :12345678
L.NASC. :CORSICO
D.NASC. :15061940
T.STUD. :PERITO CHIMICO
OCC.AT. :ANALISTA LABORATORIO
OCC.PR. :TECNICO LABORATORIO
ST.CIV. :C
NOTA 1  :OTTIMO ELEMENTO
NOTA 2  :MOLTO SCRUPOLOSO
```

Questo blocco di programma dovrà probabilmente essere modificato, se si passa ad un archivio di altra natura e diversamente strutturato.

CREAZIONE INDICE SECONDARIO
linee 1180-1235
dopo aver chiesto il numero del campo, da usare come chiave di ordinamento, per creare l'indice secondario (supponendo che possa variare tra 2 e NC), si crea l'indice secondario servendosi in memoria degli stessi vettori usati per l'indice principale. Alla fine l'indice secondario viene scritto sul dischetto con il nome INDI2, cancellando, se esiste, un precedente indice secondario.

LISTA PER INDICE SECONDARIO
linee 1240-1285
viene usato INDI2 come indice per listare il file.

CHIUSURA
linee 1290-1350
viene chiesto se l'indice principale è da ordinare; se la risposta è S, esso viene ordinato, e, comunque, viene riscritto su disco. Viene aggiornato il settore 1,0 per i dati generali del disco che sono:

- data di aggiornamento,
- numero record presenti,
- indirizzo traccia e settore ultimo settore riempito (nell'allocazione di settori per aggiungere record si parte da questo indirizzo).

Il programma termina segnalando sul video il numero dei record presenti nell'archivio.

ORDINAMENTO CRESCENTE
linee 1355-1420
è la routine di ordinamento a bolle. Essa ordina in senso crescente il vettore C$(N), trascinandosi dietro nell'ordinamento i due vettori T%(N) e S%(N). Se il numero dei record è abbastanza elevato, tale routine risulta piuttosto lenta (vedi gli esempi di ordinamento nel primo volume), ma essa può essere facilmente sostituita con una più efficiente.

ORDINAMENTO DECRESCENTE
linee 1425-1490
è la routine di ordinamento a bolle, ma ordina in senso decrescente. Vale quanto detto per la routine precedente.

INGPESSO DATI
linee 250-310
richiede i dati per un record, rende i campi della lunghezza fissa stabilita, chiede conferma dei dati ricevuti e consente di modificarli. Per uscire dalla routine senza fornire dati, basta rispondere con il carattere "$" alla richiesta del cognome. Se il record è stato ricevuto la routine ritorna lo switch W=0, mentre se il record non è stato ricevuto ritorna lo switch W=1.
La lunghezza dei campi (linee 295-310) è importante, infatti i record devono mantenere i campi della lunghezza stabilita. Qualora si desideri accedere a un particolare campo nel record, si deve poter porre il puntatore al valore previsto; se i campi non sono stati scritti come si deve, si rischiano grossi pasticci.

SCRITTURA INDICE PRINCIPALE
linee 315-350
va a scrivere il file sequenziale INDI1 su disco.

SCRITTURA NEL BUFFER
linee 355-375
trasferisce il vettore dei dati nel buffer.

ALLOCA TRACCIA E SETTORE
linee 380-415
alloca un settore, sistemando la BAM.
Non viene controllato se EN=65 e ET=0; si suppone di lavorare con un numero di record compatibile con la capacità del dischetto.

PUNTATORE NEL BUFFER
linee 420-430
pone il puntatore nella posizione 2 del buffer.

SCRITTURA RECORD
linee 435-445
scrive il buffer su disco nel settore T,S.

LETTURA RECORD
linee 450-475
legge il settore T,S nel buffer e poi trasferisce i campi nel vettore Y$(NC).

LETTURA DATI DISCO
linee 480-515
legge i dati generali del disco.

LETTURA INDICE
linee 520-545
legge in memoria INDI1. Non viene controllato l'EOF, dato che si legge un numero fisso di record.

ROUTINE ERRORE
linee 550-575
controlla gli errori relativi alle operazioni disco.

DATA DISCO
linee 580-590
richiede la data di aggiornamento.

SCRITTURA INDICE SECONDARIO
linee 595-630
scrive su disco INDI2.

RICERCA RECORD
linee 85-910
ricerca un determinato record, dopo aver chiesto: cognome, nome e occorrenza (sono ammessi record multipli). Segnala se non lo trova.

Elenco variabili e costanti

NC=14, numero campi del record
S1$, stringa descrittiva
S2$, stringa descrittiva
SP$,30 spazi, serve per aggiustare le lunghezze dei campi, deve essere lunga come il campo più lungo
CH$=CHR$(13), carattere RETURN, usato come fine campo
LIM$=CHR$(99)+CHR$(99)+CHR$(99), serve per la chiave dei record cancellati

D$(NC), descrizioni campi record
Y$(NC), campi dati
L(NC), lunghezze dei campi dati
K,J,L variabili di controllo
N, numero record
R$, variabile per risposta
X, variabile di comodo
G1$,M1$,A1$, data precedente
C$(N), vettore per chiave indice
T%(N), vettore per numero traccia
S%(N), vettore per numero settore
EN,EM$,ET,ES, variabili per routine errore
T,S, indirizzo traccia e settore
G$,M$,A$, data aggiornamento
N$, nome disco e archivio
W, switch per lettura dati
X$, variabile per risposte
TT,SS, indirizzo settore precedente
M, numero record cancellati

Il programma a volte invia dei messaggi sul video e non chiede risposta, per proseguire devi premere un tasto. Se viene chiesta la risposta affermativa, devi premere S.

Le operazioni disco vengono svolte usando i comandi U1 e U2.

Vediamo ora le critiche al programma:

- non abbiamo dedicato particolare cura ai quadri video,
- non abbiamo dedicato particolare cura alla stesura dei listati,
- non abbiamo programmato liste parziali dell'archivio,
- il programma dopo ogni funzione termina, per proseguire devi dare ancora RUN,
- SUL DISCHETTO DEI DATI NON PUO' ESSERE ESEGUITA LA FUNZIONE VALIDATE, PENA LA PERDITA DEI DATI.

Prima di chiudere, facciamo un pò di conti, cioè vediamo quali possono essere le dimensioni di un archivio gestito dal programma ARCHIRANDOM.

Il programma occupa, prima di dare RUN, 7397 byte, quindi, restano liberi per le variabili 38911-7397=31514 byte.
Il dischetto ha 664 settori disponibili. Un settore è occupato per i dati generali del disco. Per ogni record vengono occupati:

- 1 settore per il record,
- circa 43 byte per il record di INDI1,

• circa 35 byte per il record di INDI2.

Supponiamo di voler creare un archivio di 500 record; sono necessari:

• 500 settori per i record principali,

• 500*43=21500 byte per INDI1 in memoria, che diventano circa 80 settori sul dischetto,

• 500*35=17500 byte per INDI2 in memoria, ma questi non contano perchè si sovrappongono a quelli usati per INDI1; essi, però, contano sul dischetto per circa 70 settori.

In sostanza sul dischetto si occupano 1+500+80+70=651 settori, mentre in memoria occorrono circa 20000 byte per gli indici più le altre variabili. Sembra possibile arrivare a 500 record, ma devi tener presente che diventano molto lenti gli ordinamenti degli indici.Se vuoi gestire in modo sequenziale un file di questo tipo, devi leggerlo seguendo, record dopo record, il file indice INDI1.

3.11 Esempio di archivio RANDOM/USER

In questo paragrafo mostriamo come si può ovviare all'inconveniente che abbiamo puntualizzato per l'esempio precedente, cioè il fatto della non possibilità di usare il comando VALIDATE per un dischetto che contenga registrazioni RANDOM.

Procediamo in questo modo:

• prepariamo un programma, di nome INIZIO, che serve solo per inizializzare l'archivio,

• esso crea un file sequenziale di tipo USER (USR), per distinguerlo dai normali sequenziali di tipo SEQ, registrando record logici esattamente di 254 caratteri, compresi i caratteri di fine campo (RETURN),

• il primo record logico contiene i dati generali del disco, completati con campi pieni di spazi per raggiungere 254 caratteri,

• dopo, ogni record logico è formato dagli stessi campi del record logico del file RANDOM, in modo da coprire completamente il settore, solo che i campi sono tutti formati dal carattere CHR$(99) seguito, eventualmente, a seconda della lunghezza, da spazi (codice ASCII 32),

• il sistema concatena i record usando i primi due caratteri di ogni settore,

• il file di tipo USR viene registrato nella directory con il nome che gli è stato assegnato, che è anche il nome assegnato al disco dati in fase di formattazione,

• il programma chiede a priori quanti record si vogliono, tale numero va deciso in fase di creazione del file, al limite restano poi dei record vuoti,

• dopo la chiusura del file di tipo USR, il programma va a leggere nella directory l'indirizzo assegnato al primo record del file,

• il programma comincia a scrivere su disco il primo record del file indice INDI1, lasciando in bianco la chiave e registrando l'indirizzo di traccia e settore, letto nella directory, questo primo record reca l'indirizzo del settore dove sono registrati i dati generali del disco,

• il programma legge in modo diretto (RANDOM) il primo blocco e scrive su disco il secondo record del file INDI1 (che si riferisce al primo record del file random), lasciando in bianco la chiave, ma registrando l'indirizzo di traccia e settore, e, procede così per il numero dei record che sono stati registrati nel file USR,

• alla fine sul discetto è registrato un file di tipo USR, che dopo verrà usato come RANDOM, ma senza dovere più nè allocare nè liberare blocchi,

• inoltre sul dischetto è registrato INDI1, con le chiavi in bianco e gli indirizzi registrati,

• per riconoscere i bloccchi liberi serve la chiave di INDI1, contenente solo il carattere CHR$(99), seguito da spazi,

• il disco dati può essere usato da un programma ottenuto da ARCHIRANDOM, modificandolo opportunamente,

• il dischetto dei dati può subire la funzione VALIDATE senza danni,

• qualora la struttura del record del file random non arrivi a coprire 254 caratteri, si aggiunge in fondo un campo di riempimento, infatti per ottenere che il file sequenziale usi un settore per ogni record logico, questo deve essere esattamente di 254 caratteri,

• quando il sistema crea un file sequenziale (anche USR), esso salta automaticamente la traccia 18.

Per migliorare ulteriormente il nostro programma di archivio ci mettiamo nelle condizioni di non doverlo modificare, come suggerito nel paragrafo precedente, se cambiamo la struttura del record. Per semplicità continuiamo a ragionare in termini di record logico che occupa un intero settore. Possiamo procedere così:

• prepariamo un programma di strutturazione dell'archivio, di nome PRESTRUTT,

• esso ci chiede:

1) il numero dei campi del record
2) la lunghezza e il nome di ogni campo,

• controlla che la somma delle lunghezze dei campi, compresi i caratteri di fine campo, non superi 254,

• scrive un file sequenziale con tutte le informazioni sulla struttura dell'archivio,

• tale file sequenziale, che chiamiamo STRUTTURA, viene letto, sia dal programma di inizializzazione, INIZIO, che dal programma di gestione dell'archivio, ARANDOMUSER, e fornisce i dati necessari per lavorare.

Segue il listato del programma PRESTRUTT.

```
5 REM PRESTRUTT
10 S1$="LA SOMMA DELLE LUNGHEZZE SUPERA 254"
15 S2$="MODIFICA QUALCHE CAMPO"
20 PRINT"■DEFINIZIONE STRUTTURA RECORD■"
25 REM CHIEDE NUMERO CAMPI MINORE DI 20
30 INPUT"NOME FILE: ";N$
35 INPUT"QUANTI CAMPI: ";NC
40 IFNC<0ORNC>20THEN20
45 PRINT"■NOMI E LUNGHEZZE CAMPI"
50 PRINT"OGNI CAMPO MINORE DI 80 CARATT."
55 PRINT"OGNI NOME MASSIMO 10 CARATTERI"
60 DIMD$(NC),L(NC)
65 FORK=1TONC
70 PRINTK;"NOME: ";:INPUTD$(K)
75 INPUT"LUNGHEZZA: ";L(K)
80 NEXTK
85 PRINT"■CONTROLLO CAMPI"
90 FORK=1TONC
95 IFLEN(D$(K))>10THEND$(K)=LEFT$(D$(K),10)
100 IFL(K)>79THENL(K)=79
105 PRINTK;D$(K);" ";L(K)
110 NEXTK
115 PRINT"■CONFERMI S/N ";:INPUTR$
120 IFR$="S"THEN145
125 INPUT"QUALE CAMPO: ";X
130 IFX<0ORX>NCTHEN125
135 INPUT"CAMPO: ";D$(X)
140 INPUT"LUNGHEZZA: ";L(X):GOTO85
145 REM CALCOLA LUNGHEZZA RECORD
150 S=NC:FORK=1TONC
155 S=S+L(K):NEXTK
160 IFS>254THENPRINTS1$:PRINTS2$:GOTO90
165 PRINT"NOME FILE: ";N$
170 PRINT"LUNGHEZZA RECORD: ";S
175 OPEN15,8,15,"I"
180 OPEN2,8,2,"STRUTTURA,S,W"
185 PRINT#2,N$
190 PRINT#2,NC
195 FORK=1TONC:PRINT#2,D$(K):NEXTK
200 FORK=1TONC:PRINT#2,L(K):NEXTK
205 CLOSE2
210 PRINT"■VERIFICA"
```

```
215 OPEN2,8,2,"STRUTTURA,S,R"
220 INPUT#2,A$
225 PRINT"NOME FILE: ";A$
230 INPUT#2,X
235 PRINT"NUMERO CAMPI: "X
240 FORK=1TOX:INPUT#2,D$(K):NEXTK
245 FORK=1TOX:INPUT#2,L(K):NEXTK
250 CLOSE2
255 FORK=1TOX:PRINTK;D$(K);L(K):NEXTK
260 STOP
```

Il programma chiede:

- il nome da assegnare al file archivio, linea 30,
- il numero dei campi e controlla che tale numero non superi 20 (per fare entrare la descrizione nel video, un campo per linea), linee 35-40,
- il nome e la lunghezza di ogni campo, con possibilità di correzione, linee 45-140,
- se la somma delle lunghezze, compresi i caratteri di fine campo, supera 254, chiede di modificare qualche campo, linee 145-160
- taglia le descrizioni dei campi a 10 caratteri, se più lunghi, linea 95,
- taglia la lunghezza dei campi a 79, se maggiore, linea 100,
- se tutto va bene, scrivi sul dischetto dei programmi (che quindi non deve essere protetto da scrittura), il file di nome STRUTTURA, di tipo SEQ, linee 175-205,
- il file STRUTTURA contiene:
    - nome del file,
    - numero dei campi del record,
    - nome di ogni campo,
    - lunghezza di ogni campo.

Alla fine il programma ripropone, per controllo, sul video il contenuto del file STRUTTURA generato, linee 210-260. Il file STRUTTURA viene memorizzato sul disco del programma.

Vediamo ora come è stato costruito il programma INIZIO:

- legge inizialmente i dati del file STRUTTURA, mostra sul video il nome del file e il numero dei campi, poi chiede il numero N dei record da registrare per il file, ed elenca i nomi e le lunghezze dei campi, linee 10-105,
- controlla che la somma delle lunghezze non superi 254, se supera si ha uno STOP, linee 110-120,

• prepara i campi dati, CHR$(99)+spazi, linee 125-145,

• se la somma delle lunghezze è minore di 254, prepara un campo di lunghezza opportuna per riempire il settore, linee 150-170,

• chiede di montare un disco dati nuovo da formattare e preparare per l'archivio, linee 175-190,

• formatta il dischetto dati e apre un file sequenziale con il nome fornito dal file STRUTTURA e il tipo USR, linee 191-210,

• comincia a scrivere sul file il primo record relativo ai dati generali del disco, linee 215-240, che sono:

- nome file,
- data,
- numero N di record previsto (massimo),
- numero record realmente esistente (inizialmente 0),

• scrive il numero N di record richiesto, seguendo il tracciato rilevato dal file STRUTTURA, ma scrivendo campi formati da CHR$(99) seguito da spazi (o solo da CHR$(99)), linee 245-270,

• chiude il file, che rimane registrato nella directory, con tutti i settori concatenati usando i primi due byte, linea 275,

• il file occupa N+1 blocchi,

• apre un canale per lettura diretta, il canale 2, linee 280-295,

• legge in modo diretto (random) il settore 18,1 (primo della dyrectory) e, con il puntatore a 3, va a leggere l'indirizzo del primo settore usato, tale indirizzo è quello del blocco usato per i dati generali del disco, e chiude il canale 2, linee 300-35,

• apre il file INDI1 sul canale 3, e un canale ad accesso diretto il 2, linee 340-380,

• scrive il primo record di INDI1, quello relativo ai dati generali del disco, linee 385-390,

• legge in modo diretto gli N settori concatenati, prelevando dalle prime due posizioni di ogni blocco l'indirizzo di traccia e settore, e scrive il corrispondente record di INDI1, linee 395-430,

• i campi chiave di INDI1 contengono la somma dei primi due campi del record (iniziano con CHR$(99)),

• chiude i file, linee 435-440,

• esegue la VALIDATE del disco, linea 443.

```
5 REM INIZIO
10 REM DIMENSIONA COSTANTI E VARIABILI
15 REM LEGGE FILE STRUTTURA
20 CH$=CHR$(13)
25 S1$=" ":FORK=1TO39:S1$=S1$+" ":NEXTK
```

```
30 S2$="                "
35 OPEN15,8,15,"I"
40 OPEN2,8,2,"STRUTTURA,S,R":GOSUB445
45 INPUT#2,N$,NC:GOSUB445
50 N$=LEFT$(N$+S1$,16)
55 PRINT"DATI FILE STRUTTURA"
60 PRINT"FILE: ";N$;" NUMERO CAMPI: ";NC
65 INPUT"QUANTI RECORD: ";N
70 IFN>500THEN STOP
75 NX$=STR$(N)
80 NX$=RIGHT$(S2$+NX$,6)
85 DIMD$(NC),L(NC),Y$(NC)
90 FORK=1TONC:INPUT#2,D$(K):NEXTK
95 FORK=1TONC:INPUT#2,L(K):NEXTK
100 CLOSE2
105 FORK=1TONC:PRINTD$(K);"    ";L(K):NEXTK
110 SR=NC:FORK=1TONC:SR=SR+L(K):NEXTK
115 PRINT"LUNGHEZZA RECORD: ";SR
120 IFSR>254THENPRINT"NON POSSO PROSEGUIRE":STOP
125 REM CAMPI DATI CON CHR$(99) + SPAZI
130 FORK=1TONC:Y$(K)=CHR$(99):IFL(K)=1THEN145
135 FORJ=1TOL(K)-1
140 Y$(K)=Y$(K)+CHR$(32):NEXTJ
145 NEXTK
150 IFSR=254THEN175
155 REM CREA CAMPO FINALE SE SR<254
160 IFSR=253THENDN$="":GOTO175
165 DN$=CHR$(99):FORK=1TO253-SR-1
170 DN$=DN$+CHR$(32):NEXTK
175 REM CREA FILE USER
180 PRINT"MONTA DISCO DATI"
185 PRINT"PREMI UN TASTO PER PROSEGUIRE"
190 R$="":GETR$:IFR$=""THEN190
191 PRINT"PAZIENZA, ATTENDI, STO PREPARANDO"
192 PRINT"IL DISCO DATI"
195 CLOSE15:OPEN15,8,15,"N0:"+N$+",99"
200 REM RIEMPIE RECORD
205 REM CON CAMPI CHE INIZIANO CON CHR$(99)
210 OPEN2,8,2,N$+",U,W":GOSUB445
215 REM SCRIVE PRIMO RECORD PER DATI DISCO
220 REM OCCUPANDO UN INTERO SETTORE
225 PRINT#2,N$CH$"GGMMAA"CH$NX$CH$"0"CH$" "CH$" "CH$;
```

```
230 GOSUB445
235 PRINT#2,S1$CH$S1$CH$S1$CH$S1$CH$S1$CH$S2$
240 GOSUB445
245 FORK=1TON
250 FORJ=1TONC
255 PRINT#2,Y$(J);CH$;
260 GOSUB445
265 NEXTJ
266 IFSR<254THENPRINT#2,DN$
270 NEXTK
275 CLOSE2
280 REM CREA FILE INDICE
285 REM INDIRIZZO PRIMO SETTORE DIRECTORY
290 CLOSE15:OPEN15,8,15,"I"
295 OPEN2,8,2,"#":GOSUB445
300 REM LEGGE SETTORE 18,1
305 PRINT#15,"U1:2,0,18,1":GOSUB445
310 REM PUNTATORE A TRACCIA E SETTORE
315 PRINT#15,"B-P:2,3":GOSUB445
320 T$="":S$="":GET#2,T$,S$
325 T$=T$+CHR$(0):S$=S$+CHR$(0)
330 CLOSE2
335 T=ASC(T$):S=ASC(S$)
340 PRINT"PRIMO BLOCCO"
345 PRINT"TRACCIA: ";T;"  SETTORE: ";S
350 PRINT"PREMI UN TASTO PER PROSEGUIRE"
355 R$="":GETR$:IFR$=""THEN355
360 REM GENERA INDICE
365 REM LEGGENDO FILE USR COME RANDOM
370 CLOSE15:OPEN15,8,15,"I"
375 OPEN2,8,2,"#":GOSUB445
380 OPEN3,8,3,"INDI1,S,W":GOSUB445
385 REM SCRIVE PRIMO RECORD INDI1
390 PRINT#3,Y$(1)+Y$(2);CH$;T;CH$;S:GOSUB445
395 REM SCRIVE ALTRI N RECORD
400 FORK=1TON
405 PRINT#15,"U1:2,0";T;S:GOSUB445
410 PRINT#15,"B-P:2,0":GOSUB445
415 GET#2,T$,S$:GOSUB445
416 T=ASC(T$+CHR$(0)):S=ASC(S$+CHR$(0))
420 PRINTT,S
```

```
425 PRINT#3,Y$(1)+Y$(2);CH$;T;CH$;S:GOSUB445
430 NEXTK
435 PRINT"IL DISCO DATI E' PRONTO"
440 CLOSE2:CLOSE3:CLOSE15
443 OPEN15,8,15,"V":CLOSE15:STOP
445 INPUT#15,EN,EM$,ET,ES
450 IFEN=0THENRETURN
455 PRINT"ERRORE DISCO":PRINTEN,EM$,ET,ES
460 STOP
```

Il dischetto dati, prodotto da INIZIO, in collaborazione con PRESTRUTT, è sicuro, cioè tutto è registrato nella directory.

Puoi provare a listare INDI1 con il programma CONTRINDI1, che segue; così puoi vedere come il sistema usa i settori del dischetto, partendo da 17,0 e arrivando alla traccia 1, per proseguire poi dalla traccia 19.

```
10 REM CONTRINDI1
20 OPEN15,8,15,"I"
30 OPEN2,8,2,"INDI1,S,R"
35 OPEN4,4:PRINT#4,"CONTROLLO INDI1":PRINT#4
40 K=0:I=0
45 INPUT#2,A$,T,S:TS=ST:K=K+1
50 PRINT#4,K;T;S;"  ";
60 I=I+1:IFI=5THENPRINT#4:I=0
70 IFTS=64THENPRINT#4:CLOSE2:CLOSE4:CLOSE15:STOP
80 GOTO45
```

Noi abbiamo provato a generare un file dati, di nome PROVE, con 500 record, la directory del dischetto è riportata qui:

```
0 [illegible]
501   "PROVE              " USR
90    "INDI1"               SEQ
73 BLOCKS FREE.
```

come puoi vedere, con 500 record restano pochi blocchi per l'indice secondario; i nostri conti precedenti erano stati troppo ottimisti, può funzionare se si crea un indice secondario per un campo chiave di pochi caratteri.Si potrebbe risparmiare spazio su disco comprimendo l'indice, cioè riducendo il record logico da 3 campi a un campo, senza caratteri separatori, e memorizzando il numero di traccia e il numero di settore come una sola coppia di byte (CHR$) in coda alla chiave. Naturalmente gli altri programmi che usano INDI1 dovrebbero tenere conto di questa struttura. Noi abbiamo memorizzato traccia e settore come numeri, occupando più spazio del necessario.

Puoi, servendoti del programma di utilità DISPLAY T&S, stampare il contenuto di qualche blocco, per controllare come ha lavorato il programma INIZIO.

Ora dobbiamo vedere come trasformare il programma ARCHIRANDOM, esposto nel paragrafo precedente, per farlo lavorare su un dischetto dati preparato con i programmi PRESTRUTT e INIZIO. Il nuovo programma per trattare i file RANDOM/USER si chiama ARANDOMUSER.

Le principali differenze rispetto a ARCHIRANDOM sono le seguenti:

- il settore dei dati generali del disco non è più quello di indirizzo 1,0, ma il suo indirizzo si trova leggendo il primo record del file INDI1,
- il file INDI1 deve sempre essere caricato nei tre vettori dell'indice, qualunque scelta si faccia, a partire dal secondo record (il file ha N+1 record),
- la fase di inizializzazione ex-novo dell'arcrchivio sparisce dal programma,
- tra i dati generali del disco ne figurano uno in più e due in meno, infatti si ha il numero N dei record previsti al massimo, e il numero K dei record effettivamente presenti, mentre non servono più la traccia e il settore dell'ultimo record scritto,
- il file indice INDI1 deve sempre essere ordinato e riscritto tutto, altrimenti si perdono gli indirizzi dei record,
- nell'ordinamento del file indice in memoria, i record non usati, che iniziano con CHR$(99), vanno in fondo, ma restano,
- per cancellare un record, si pone CHR$(99) seguito da spazi nella chiave di INDI1, per uniformarsi agli altri record vuoti,
- il file INDI2 si riferisce solo ai record effettivamente presenti K, ma per operare correttamente si deve lavorare con un indice principale ordinato,
- quando si aggiunge un nuovo record, si deve cercare nell'indice la prima chiave che inizia con CHR$(99), leggere il corrispondente settore e riscriverlo con i nuovi dati; non si devono infatti perdere i primi due caratteri di concatenamento.

Segue il listato di ARANDOMUSER.

```
5 REM ARANDOMUSER
10 REM-----------------------------------
15 REM DEFINIZIONE COSTANTI E VARIABILI
20 REM-----------------------------------
25 S1$="DATA ULTIMO AGG.      "
30 S2$="FINITO SPAZIO ASSEGNATO"
35 SP$="                              "
40 CH$=CHR$(13):LIM$=CHR$(99)+CHR$(32)+CHR$(32)
45 REM **LEGGE DATI DA FILE STRUTTURA**
50 REM ******************************
55 OPEN15,8,15,"I"
57 PRINT"{CLR}{DOWN}{DOWN}"TAB(8)"MONTA DISCO FILE STRUTTURA"
58 GOSUB610
60 OPEN2,8,2,"STRUTTURA,S,R":GOSUB525
65 INPUT#2,N$,NC:GOSUB525
70 DIMD$(NC):DIMY$(NC):DIML(NC)
75 FORK=1TONC:INPUT#2,D$(K):NEXTK
80 FORK=1TONC:INPUT#2,L(K):NEXTK
85 CLOSE2
90 PRINT"{CLR}{DOWN}{DOWN}"TAB(10)"MONTA DISCO DATI"
95 GOSUB610
100 CLOSE15:OPEN15,8,15,"I"
105 REM **LEGGE DATI DISCO**
110 REM *******************
115 OPEN10,8,10,"INDI1,S,R":GOSUB525
120 REM **SETTORE DATI GENERALI**
125 REM ***********************
130 INPUT#10,A$,TD,SD
135 GOSUB455
140 REM **DIMENSIONAMENTI PER INDICE**
145 REM ***************************
150 DIMC$(N),T%(N),S%(N)
155 REM **LETTURA INDICE**
160 REM ****************
165 GOSUB510:GOSUB555
170 REM **PRESENTAZIONE MENU' E SCELTE**
175 REM *****************************
180 PRINT"{CLR}{DOWN}{DOWN}";TAB(10);"GESTIONE ARCHIVIO":PRINT
185 PRINTTAB(10)"1=AGGIORNAMENTO"
190 PRINTTAB(10)"2=LISTA PRINCIPALE"
195 PRINTTAB(10)"3=CREAZ.IND.SEC."
200 PRINTTAB(10)"4=LISTA SECONDARIA"
```

```
205 PRINTTAB(10)"9=FINE"
210 INPUT"COSA SCEGLI ";X
215 IFX<1ORX>4ANDX<>9THENPRINT"]";:GOTO210
220 PRINT:PRINT"PRESENTI ";K;" RECORD"
225 PRINTS1$;G1$;"/";M1$;"/";A1$
226 GOSUB610
230 IFX=9THEN1230
235 ONXGOTO705,1035,1120,1180
240 STOP
245 REM **INGRESSO DATI**
250 REM ****************
255 PRINT"]INGRESSO DATI"
260 FORJ=1TONC:Y$(J)="":NEXTJ
265 PRINT"PER USCIRE $ PER COGNOME";
270 PRINT"▮SE MANCANO DATIPREMI SOLO RETURN"
275 PRINTD$(1);:INPUTY$(1)
280 IFY$(1)="$"THENW=1:RETURN
285 FORJ=2TONC:PRINTD$(J);:INPUTY$(J):NEXTJ
290 REM **SISTEMA LUNGHEZZA DATI**
295 REM *************************
300 FORJ=1TONC:Y$(J)=LEFT$(Y$(J)+SP$,L(J))
305 NEXTJ:GOSUB900:RETURN
310 REM **SCRITTURA INDICE PRINCIPALE**
315 REM ******************************
320 PRINT#15,"S0:INDI1":PRINT#15,"I"
325 OPEN10,8,10,"INDI1,S,W":GOSUB525
330 PRINT#10,LIM$;CH$;TD;CH$;SD;CH$;:GOSUB525
335 FORJ=1TON
340 PRINT#10,C$(J);CH$;T%(J);CH$;S%(J);CH$;
345 GOSUB525:NEXTJ
350 CLOSE10:RETURN
355 REM **SCRITTURA NEL BUFFER**
360 REM **********************
365 FORJ=1TONC
370 PRINT#11,Y$(J);CH$;:GOSUB525
375 NEXTJ:RETURN
380 REM **LEGGI SETTORE NEL BUFFER**
385 REM ***************************
390 PRINT#15,"U1:11,0";T;S:GOSUB525:RETURN
395 REM **PUNTATORE NEL BUFFER**
400 REM **********************
405 PRINT#15,"B-P:"11;2:GOSUB525:RETURN
```

```
410 REM **SCRITTURA RECORD**
415 REM ********************
420 PRINT#15,"U2:"11;0;T;S:GOSUB525:RETURN
425 REM **LETTURA RECORD**
430 REM ******************
435 PRINT#15,"I":OPEN11,8,11,"#":GOSUB525
440 PRINT#15,"U1:"11;0;T;S:GOSUB525:GOSUB405
445 FORJ=1TONC:INPUT#11,Y$(J):GOSUB525
450 NEXTJ:CLOSE11:RETURN
455 REM **LETT.DATI DISCO**
460 REM *******************
465 OPEN11,8,11,"#":GOSUB525
470 PRINT#15,"U1:"11;0;TD;SD:GOSUB525
475 GOSUB395:INPUT#11,N$,R$,N,K:GOSUB525
480 G1$=LEFT$(R$,2):M1$=MID$(R$,3,2)
485 A1$=RIGHT$(R$,2):CLOSE11:RETURN
490 REM **LETT. INDICE**
495 REM ****************
496 CLOSE10:OPEN10,8,10,"INDI1,S,R":GOSUB525
497 INPUT#10,A$,TD,SD
500 REM **DA SECONDO RECORD**
505 REM *********************
510 FORJ=1TON
515 INPUT#10,C$(J),T%(J),S%(J):GOSUB525
520 NEXTJ:CLOSE10:RETURN
525 REM **ROUTINE ERRORE**
530 REM ******************
535 INPUT#15,EN,EM$,ET,ES
540 IFEN=0THENRETURN
545 PRINT"█ERRORE DISCO"
550 PRINTEN,EM$,ET,ES:CLOSE15:STOP
555 REM **DATA DISCO**
560 REM **************
565 PRINT"DATA PER DISCO"
570 INPUT"   GG,MM,AA████████████";G$,M$,A$:RETURN
575 REM **SCRITT. IND. SEC.**
580 REM ********************
585 PRINT#15,"S:INDI2":PRINT#15,"I"
590 OPEN10,8,10,"INDI2,S,W"
595 FORJ=1TOK
600 PRINT#10,C$(J);CH$;T%(J);CH$;S%(J);CH$;
605 GOSUB525:NEXTJ:CLOSE10:RETURN
```

```
608 REM **ATTESA TASTO**
609 REM ****************
610 R$="":GETR$:IFR$=""THEN610
615 RETURN
620 REM **RICERCA POSTO LIBERO**
625 REM ************************
630 FORI=1TON
635 IFLEFT$(C$(I),1)<>CHR$(99)THEN645
640 T=T%(I):S=S%(I):JJ=I:I=N
645 NEXTI:RETURN
650 REM **AGGIUNTA NUOVI RECORD**
655 REM *************************
660 GOSUB245:GOSUB620
665 IFW=1THENK=K-1:CLOSE11:GOTO1230
670 GOSUB380:GOSUB395:GOSUB355
675 REM **AGGIORNAMENTO INDICE**
680 REM ************************
685 C$(JJ)=Y$(1)+Y$(2)
690 GOSUB410:K=K+1
695 IFK>NTHENK=K-1:PRINTS2$:CLOSE11:GOTO1230
700 GOTO660
705 REM---------------
710 REM AGGIORNAMENTO
715 REM---------------
720 PRINTTAB(10);"□■■AGGIORNAMENTO ARCHIVIO"
725 PRINT:PRINTTAB(10);"1=CORREZIONE"
730 PRINTTAB(10);"2=AGGIUNTA ELEM."
735 PRINTTAB(10);"3=CANCELL.ELEM."
740 PRINTTAB(10);"9=FINE"
745 INPUT"COSA SCEGLI ";X:IFX=9THEN1230
750 IFX<1ORX>3THEN720
755 IFX=2THEN950
760 IFX=3THEN990
765 GOTO850
770 REM **RICERCA RECORD**
775 REM ******************
780 PRINT"□";D$(1);:INPUTY$(1):W=0
785 IFY$(1)="$"THENW=1:RETURN
790 PRINTD$(2);:INPUTY$(2)
795 Y$(1)=LEFT$(Y$(1)+SP$,L(1))
800 Y$(2)=LEFT$(Y$(2)+SP$,L(2))
```

```
805 INPUT"QUALE OCCORRENZA ";X
810 IFX<=0THENPRINT"⌉";:GOTO805
815 FORJ=1TON:IFC$(J)=Y$(1)+Y$(2)THEN835
820 NEXTJ
825 J=J-1:PRINT"NON TROVATO ";Y$(1);"  ";Y$(2)
830 GOSUB610:GOTO770
835 IFX<>1THENX=X-1:GOTO820
840 T=T%(J):S=S%(J)
845 I=J:J=N:NEXTJ:RETURN
850 REM **CORREZIONE**
855 REM **************
860 GOSUB770:IFW=1THEN720
865 GOSUB425
870 REM **VA A MODIFICA DATI**
875 REM **********************
880 GOSUB900
885 PRINT#15,"I":OPEN11,8,11,"#":GOSUB395:GOSUB355
890 C$(I)=Y$(1)+Y$(2):T%(I)=T:S%(I)=S
895 GOSUB410:CLOSE11:GOTO850
900 REM **CONTROLLO DATI E MODIFICHE**
905 REM ******************************
910 PRINT"⊐CONTROLLO DATI E MODIFICHE"
915 FORJ=1TONC:PRINTJ;" ";D$(J);" ";Y$(J):NEXTJ
920 INPUT"⌧TUTTO BENE S/N ";X$:IFX$="S"THENRETURN
925 PRINT"QUALE CAMPO (0 USCITA)";:INPUTX
930 IFX=0THENRETURN
935 IFX>NCORX<1THENPRINT"⌉⌉":GOTO920
940 INPUTY$(X):Y$(X)=LEFT$(Y$(X)+SP$,L(X))
945 GOTO910
950 REM **AGGIUNTA**
955 REM ************
960 OPEN11,8,11,"#":GOSUB525
965 PRINT"⊐AGGIUNTA NUOVI RECORD"
970 PRINT"PRESENTI ";K;" RECORD"
975 PRINT"PUOI AGGIUNGERE ";N-K;" RECORD"
980 GOSUB610
985 W=0:K=K+1:GOTO660
990 REM **CANCELLAZIONE**
```

```
995 REM ****************
1000 M=0
1005 GOSUB770:IFW=1THEN1025
1010 X=L(1)+L(2):C$(I)=LEFT$(LIM$+SP$+SP$,X)
1015 M=M+1
1020 GOTO1005
1025 REM SIST. INDICE
1030 GOSUB1295:K=K-M:GOTO1255
1035 REM------------
1040 REM LISTA FILE
1045 REM------------
1050 T$="LISTA PER INDICE PRIM."
1055 PRINT"ACCENDI STAMPANTE       PREMI UN TASTO"
1060 GOSUB610
1065 PRINT"♥";T$:OPEN4,4
1070 PRINT#4:PRINT#4:PRINT#4,T$
1075 FORJ=1TO10:PRINT#4:NEXTJ
1080 L=2:FORI=1TOK:T=T%(I):S=S%(I):GOSUB425
1085 PRINT#4,Y$(1);"   ";Y$(2)
1090 PRINT#4,Y$(3);"   ";Y$(4);"   ";Y$(5)
1095 FORM=6TONC:PRINT#4,D$(M)":  "Y$(M):NEXTM
1100 PRINT#4:PRINT#4
1105 IFL=5THENPRINT#4:L=0
1110 L=L+1:NEXTI
1115 CLOSE4:GOTO1280
1120 REM--------------------------------
1125 REM CREAZIONE INDICE SECONDARIO
1130 REM--------------------------------
1135 PRINT"♥INDICE SECONDARIO"
1140 INPUT"■QUALE CAMPO ";X$
1145 X=VAL(X$):IFX<2ORX>NCTHENGOTO1140
1155 FORI=1TOK:T=T%(I):S=S%(I):GOSUB425
1160 C$(I)=Y$(X):NEXTI:GOSUB1365
1165 GOSUB575
1170 PRINT"FINITO IND. SEC."
1175 GOTO1280
1180 REM--------------------------------
1185 REM LISTA PER INDICE SECONDARIO
1190 REM--------------------------------
```

```
1195 PRINT"LISTA IND. SEC."
1200 OPEN10,8,10,"INDI2,S,R":GOSUB525
1205 FORJ=1TOK:INPUT#10,C$(J),T%(J),S%(J)
1210 GOSUB525:NEXTJ
1215 CLOSE10
1220 T$="LISTA IND. SEC. "
1225 GOTO1055
1230 REM **CHIUSURA**
1235 REM ************
1240 PRINT"CHIUSURA ARCHIVIO"
1245 PRINT"IND. PRINC. DA ORDIN. S/N ":INPUTR$
1250 IFR$="S"THENGOSUB1295
1255 GOSUB310:OPEN11,8,11,"#":GOSUB525
1260 PRINT#15,"B-P:"11,2
1265 PRINT#11,N$CH$G$M$A$CH$;N;CH$;K;CH$;
1270 GOSUB525
1275 PRINT#15,"U2:"11;0;TD;SD:GOSUB525:CLOSE11
1280 PRINT"FINITO AGGIORNAMENTO"
1285 PRINT"SONO PRESENTI ";K;" RECORD"
1290 CLOSE15:STOP
1295 REM **ORDINAMENTO CRESCENTE**
1300 REM *************************
1305 L=N-1
1310 W=0
1315 FORJ=1TOL
1320 IFC$(J)<=C$(J+1)THEN1345
1325 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
1330 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1335 X=S%(J):S%(J)=S%(J+1):S%(J+1)=X
1340 W=1
1345 NEXTJ
1350 IFW=0THENRETURN
1355 IFL=1THENRETURN
1360 L=L-1:GOTO1310
1365 REM **ORDINAMENTO DECRESCENTE**
1370 REM ***************************
1375 L=K-1
1380 W=0
```

```
1385 FORJ=1TOL
1390 IFC$(J)>=C$(J+1)THEN1415
1395 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
1400 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1405 X=S%(J):S%(J)=S%(J+1):S%(J+1)=X
1410 W=1
1415 NEXTJ
1420 IFW=0THENRETURN
1425 IFL=1THENRETURN
1430 L=L-1:GOTO1380
```

Non ripetiamo la lista delle variabili usate, infatti il programma è stato costruito modificando il precedente, e le differenze sono poche. Segnaliamo le variabili TD e SD, che contengono l'indirizzo del settore utilizzato per i dati generali del disco, e che deve essere aggiornato prima di chiudere.

Anche questo programma potrebbe essere migliorato, infatti ogni funzione termina con la chiusura, e dato che il file guida, STRUTTURA, sta sul disco dei programmi, è necessario cambiare disco dopo aver dato nuovamente il RUN. Puoi modificare la linea 1290 così:

```
1290 CLOSE15:GOTO90
```

in modo che finita una fase ricompare la domanda MONTA DISCO DATI, come misura di prudenza per verificare che è montato il disco giusto. In questo modo non vengono persi i dati letti dal file STRUTTURA, come succede scrivendo RUN, e non si deve rimontare il disco dei programmi.

Altro perfezionamento, utile se il file è formato da molti record, può essere quello di creare uno SWITCH, che viene messo a zero, quando viene letto in memoria INDI1, e viene posto a 1 se si operano aggiunte, cancellazioni o modifiche dei campi chiave dei record. Alla fine se lo switch è rimasto a zero, non è necessario nè riordinare l'indice, nè riscriverlo sul dischetto, infatti non è variato.

Con file formati da molti record risulta necessario sostituire le routine di ordinamento con altre più veloci.

Un'ultima considerazione riguardo all'indice del file, qualora i record logici fossero più corti, per esempio 127 caratteri, in modo da poterne scrivere due in un settore, nell'indice, oltre all'indirizzo del settore, dovrebbe comparire l'indicazione 1 o 2, o altro, per individuare il record. Inoltre per modificare o scrivere un record si dovrebbe sempre leggere tutto il settore , andare a modificare la parte che interessa, e poi riscrivere tutto il settore.Un file di tipo RANDOM/USER, come quello costruito da noi, può anche essere gestito come file sequenziale, senza ricorrere all'indice INDI1. La sequenza dei record dipende dal caricamento iniziale, dalle cancellazioni e dalle aggiunte.

3.12 File RELATIVI di dati

I FILE RELATIVI sono organizzati in modo tale, che, per accedere a un record logico, si deve fornire al DOS il suo numero d'ordine nel file. I record logici devono essere di lunghezza fissa. Il sistema, conoscendo la lunghezza in caratteri di un record logico e il suo numero d'ordine nel file, può, con un semplice algoritmo, determinare in quale blocco fisico inizia il record e quindi accedere ad esso, sia in lettura che in scrittura. Per poter lavorare così è necessario che i record logici siano tutti della stessa lunghezza, ma non è necessario che la loro dimensione sia un sottomultiplo di un settore. Sono consentiti record logici di lunghezza a piacere, purchè minore o uguale a 254, compresi i caratteri separatori di campo e di record. Un record logico può, come del resto nei file sequenziali, stare a cavallo di due settori (spanned).

Il sistema, per gestire i file relativi, usa la tecnica che descriviamo:

- Quando crea il file costruisce un suo indice interno, chiamato SIDE SECTOR. Esso consiste in un massimo di 6 settori, numerati logicamente da 0 a 5, che servono per contenere l'indice di tutti i blocchi fisici (settori) utilizzati per il file relativo. In ogni blocco SIDE SECTOR possono essere registrati gli indirizzi di 120 settori; per questa ragione un file relativo può occupare al massimo 120*6=720 settori. Per i nostri dischetti tale numero è sovrabbondante, dato che essi dispongono solo di 664 settori, e da questi dobbiamo toglierne 6 per i SIDE SECTOR. Il numero dei record logici gestibili non può superare 65535, infatti il numero di record logico si esprime in due byte. La struttura di un SIDE SECTOR è la seguente:

byte 0 e 1, concatenamento al settore successivo
byte 2, numero del side sector, da 0 a 5
byte 3, lunghezza del record logico, massimo 254
byte 4 e 5, traccia e settore del side sector 0
byte 6 e 7, traccia e settore del side sector 1
byte 8 e 9, traccia e settore del side sector 2
byte 10 e 11, traccia e settore del side sector 3
byte 12 e 13, traccia e settore del side sector 4
byte 14 e 15, traccia e settore del side sector 5
byte 16 e 17, indirizzo primo settore dati
byte 18 e 19, indirizzo secondo settore dati
...
...
byte 254 e 255, indirizzo centoventesimo settore dati.

- I settori usati per i dati sono, come sempre, concatenati usando i primi due byte, e ogni settore può contenere 254 caratteri di dati.

- Viene registrata una entrata nella directory, che indica il numero totale dei blocchi occupati, comprendendo i blocchi per i dati e quelli occupati per i side sector. In questa entrata sono usati 3 byte per specificare l'indirizzo del primo side sector e la lunghezza del record logico.

- Quando si crea il file, i record logici non usati sono riempiti dal sistema ponendo FFH (255 in esadecimale) nella prima posizione e zeri binari (00H) nelle successive. I record logici sono così marcati a priori.

Per gestire un file relativo il sistema usa un buffer in più, tre invece di due, dato che deve gestire anche i side sector. Diminuisce così il numero dei file che si possono trattare contemporaneamente.

Vediamo i comandi necessari per gestire i file relativi.

APERTURA DEL CANALE COMANDI:

OPEN15,8,15

come sempre.

APERTURA DEL FILE E DEL CANALE PER I DATI:

OPEN#lfn,8,sa,fn+",L,"+CHR$(LU)

dove:
lfn, numero logico del file, che noi, per abitudine, facciamo coincidere con il numero di canale sa,
8, numero della periferica,
sa, numero del canale, da 2 a 14,
fn, nome del file, che può essere preceduto da "0:" per l'unità,
L, parametro richiesto, precede la lunghezza del record logico,
LU, lungheza del record logico, passata come carattere con CHR$, minore di 255.

Nella OPEN non deve essere specificato il tipo di operazione, infatti si può sia leggere che scrivere, intercalando le operazioni.

LOCALIZZAZIONE DEL RECORD E, EVENTUALMENTE, DEL CAMPO NEL RECORD:

PRINT#15,stringa-comando

dove la stringa-comando si costruisce così:

"P"+CHR$(lfn)+CHR$(LO)+CHR$(HI)+CHR$(BI)
dove:
P, sta per puntatore
lfn, è il numero logico del file usato nella OPEN
LO e HI, sono rispettivamente il byte basso e il byte alto del numero del record; si calcola:
HI=INT(numrec/256)
LO=numrec-HI*256
BI, è il puntatore al byte nel record, cioè al primo byte di un campo: 1 per il primo campo, N per un campo che si trovi dopo i primi N-1 caratteri dall'inizio del record. Questa PRINT#15, deve esssere usata per puntare a un record e a una posizione; dopo si può procedere con PRINT#lfn, con INPUT#lfn o con GET#lfn, andando in sequenza. Si deve ridefinire una posizione se si salta altrove nel file.

PER SCRIVERE DATI:

PRINT#lfn, lista-dati

con le solite regole, già viste, relative ai separatori tra le variabili e ai separatori da registrare. Devi però fare attenzione a quanto segue:

- dopo aver posizionato il puntatore, la lista-dati che usi con la PRINT#lfn, deve comprendere tutti i campi che vuoi scrivere, uno di seguito all'altro in quel particolare record; devi cioè usare una sola PRINT per scrivere in un record,

• se usi diverse PRINT# successive, il dato viene via via scritto in record diversi, successivi a quello puntato con la PRINT#15,

• naturalmente puoi usare diverse PRINT# per scrivere in uno stesso record, ma devi riposizionare il puntatore, al record e alla posizione, prima di scrivere ogni campo.

PER LEGGERE DATI:

INPUT#lfn, lista-dati

con le solite regole riguardo alla lunghezza dei campi, al loro tipo, e ai caratteri separatori.

Per leggere carattere per carattere:

GET#lfn, lista-dati

per leggere un carattere in ogni variabile, che è bene sia di tipo stringa.

PER CHIUDERE IL FILE:

CLOSElfn e poi **CLOSE15**.

La routine di errore, in fase di preestensione del file, deve ammettere il codice di errore 50.

Fare la PREESTENSIONE DEL FILE significa aprire il file e creare tutti i record in modo DUMMY, cioè riempiendoli con caratteri di comodo. Se si fa inizialmente questa operazione, la gestione del file risulta più veloce, infatti sono creati subito tutti i side sector necessari. Per fare la preestensione, basta aprire il file e andare a scrivere l'ultimo record; infatti il sistema per poter scrivere l'ultimo record deve predisporre tutti quelli che vengono prima. I record sono predisposti dal sistema ponendo FFH come primo carattere e continuando con 00H, senza separazione tra i campi, che esso non conosce. Qualora l'ultimo blocco dati (settore) non sia occupato tutto, il sistema lo completa con record DUMMY, pur registrando nel secondo byte la reale occupazione.

Abbiamo preparato dei programmi esempio. Iniziamo con CREAREL; esso crea un file di nome PROVA REL, formato da 30 record logici di 21 caratteri ciascuno. A questo punto non interessa la suddivisione in campi del record logico. Il programma crea il file e lo preestende andando a scrivere, a partire dal fondo 30 record formati da 20 lineette più il carattere CHR$(13).

```
5 REM CREAREL
10 REM CREAZIONE FILE RELATIVO
15 REM PREESTENSIONE FILE
20 REM 30 RECORD DI 21 CARATTERI
25 REM COMPRESO RETURN FINALE
30 DR$="0":NF$=DR$+":PROVA REL,L,"
35 SA=2:LF=2:RC$="----------":RC$=RC$+RC$
40 BI=1:NR=30:LU=21
45 OPEN15,8,15,"I"
50 REM APERTURA FILE
55 OPENLF,8,SA,NF$+CHR$(LU)
60 GOSUB135
65 REM PREESTENSIONE FILE
70 REM PER NR RECORD
75 FORX=NRTO1STEP-1
80 HI=INT(X/256):LO=X-HI*256
85 C$="P"+CHR$(LF)
90 REM NUMERO RECORD (BYTE BASSO + BYTE ALTO)
95 C$=C$+CHR$(LO)+CHR$(HI)
100 REM PUNTATORE AL PRIMO BYTE DEL RECORD
105 C$=C$+CHR$(BI)
110 REM POSIZIONA IL PUNTATORE
115 PRINT#15,C$:GOSUB135
120 REM SCRIVE IL RECORD DI LINEETTE
125 PRINT#LF,RC$:GOSUB135
130 NEXTX:CLOSELF:GOSUB135:CLOSE15:END
135 INPUT#15,EN,EM$,ET,ES
140 IFEN=0OREN=50THENRETURN
145 PRINTEN;EM$;ET;ES
150 STOP:RETURN
```

Il programma opera così:

- Prepara le costanti necessarie, lf sta per numero logico del file (non si può usare LFN, dato che FN è una parola chiave). Pone sa e lf a 2, LU=21 per lunghezza record logico, RC$ contiene 20 lineette, BI=1 per puntare al primo carattere del record', NR=30 per numero di record, linee 5-40.
- Apre il canale 15 e inizializza il dischetto, linea 45.
- Apre il file PROVA REL, linee 50-60.
- Preestende il file scrivendo record di lineette. Nota come vengono svolti i

calcoli, dato che si parte dal fondo, ogni volta deve essere ricalcolato il numero del record, e deve essere eseguita l'operazione di posizionamento del puntatore. Linee 65-130.

Noi, per provare il programma, abbiamo agito così:

- formattato un dischetto dati,
- lanciato il programma CREAREL,
- listato la directory del dischetto dati dopo l'esecuzione del programma, che riportiamo:

```
0 [illegible]
4     "PROVA REL"          REL
660 BLOCKS FREE.

READY.
```

puoi vedere che il file occupa 4 blocchi, ovviamente 3 per i dati e 1 per l'unico side sector necessario,

- abbiamo caricato il programmma DCOMEFS e l'abbiamo lanciato usando come dischetto quello con il file PROVA REL, abbiamo così osservato che il file inizia nel settore 17, 0 e che il side sector si trova nel settore 17,10, come puoi vedere nella parte di risultati sotto riportati:

```
RELATIVI  RR
 160  50  65  160  160  160  160


REL    17  0 PROVA REL
 17  10  21
 0  0
 4  0
DEL    0  0
 0  0  0
 0  0
 0  0
DEL    0  0
 0  0  0
 0  0
 0  0
```

• abbiamo caricato il programmmma DISPLAY T&S, e abbiamo stampato i settori del dischetto dati che contengono il file PROVA REL, cioè il 17,0; il 17,11; il 17,1; e poi il 17,10.

Non riportiamo qui i risultati della stampa, che puoi facilmente ottenere anche tu. Ti consigliamo, però, di fare la prova, per renderti conto di quanto abbiamo esposto sui file relativi; vedrai, per esempio, i record a cavallo dei settori.

Segue il programma SCRIVIREL, con il quale si possono scrivere record nel file PROVA REL, prima creato. Questo programma scrive nei record logici voluti due campi dati, il primo di 10 caratteri e il secondo di 9, più i fine campo. Puoi scrivere il record logico che desideri; se dai due volte lo stesso numero di record, il secondo si sovrappone al primo.

```
5 REM SCRIVIREL
10 REM SCRITTURA RECORD NEL FILE RELATIVO
15 REM OGNI RECORD 2 CAMPI
20 REM NOME DI 10 CARATTERI
25 REM TELEFONO DI 9 CARATTERI
30 REM LUNGHEZZA RECORD=10+1+9+1=21
35 DR$="0":NF$=DR$+":PROVA REL,L,"
40 LU=21:SA=2:LF=2:B$="                    "
45 BI=1:NR=30
50 OPEN15,8,15,"I"
55 REM APERTURA FILE
60 OPENLF,8,SA,NF$+CHR$(LU)
65 GOSUB160
70 PRINT"NOME=* PER USCIRE":PRINT
75 INPUT"NOME(10)";N$
80 IFN$="*"THEN155
85 N$=LEFT$(N$+B$,10)
90 INPUT"TEL. (9)";T$:T$=LEFT$(T$+B$,9)
95 INPUT"RECORD ";R
100 IFR=0ORR>NRTHENPRINTCHR$(145);:GOTO95
105 REM PREPARAZIONE PUNTATORE
110 HI=INT(R/256):LO=R-HI*256
115 C$="P"+CHR$(LF)
120 C$=C$+CHR$(LO)+CHR$(HI)
125 C$=C$+CHR$(BI)
130 REM PREDISPONE PUNTATORE PER SCRIVERE
```

```
135 PRINT#15,C$:GOSUB160
140 REM SCRIVE RECORD
145 PRINT#LF,N$;CHR$(13);T$:GOSUB160
150 GOTO70
155 CLOSELF:GOSUB160:CLOSE15:END
160 REM ROUTINE ERRORE, ACCETTA COD. 50
165 INPUT#15,EN,EM$,ET,ES
170 IFEN=0OREN=50THENRETURN
175 PRINTEN;EM$;ET;ES
180 STOP:RETURN
```

Il programma opera così:

- Prepara le costanti, linee 5-45.
- Apre il canale 15 e inizializza il dischetto dati, linea 50.
- Apre il file relativo PROVA REL; si potrebbe omettere L e la lunghezza, dato che il file esiste già. Linee 60-65.
- Chiede i dati per un record e il numero del record predispone il puntatore e scrive il record, linee 70-150.
- Se si risponde con "*" al nome, chiude il file e termina.

Dopo aver fatto girare il programma puoi andare a stampare il contenuto dei blocchi, come suggerito prima, con DISPLAY T&S; vedrai i record che hai caricato.

Segue il programma LEGGIREL per leggere un record qualunque dal file PROVA REL.

```
5 REM LEGGIREL
10 REM LETTURA RECORD
15 DR$="0":NF$=DR$+":PROVA REL,L,"
20 LU=21:SA=2:LF=2
25 BI=1:NR=30
30 OPEN15,8,15,"I"
35 REM APRE FILE
40 OPENLF,8,SA,NF$+CHR$(LU)
45 GOSUB135
50 PRINT"RECORD=0 PER USCIRE":PRINT
55 INPUT"RECORD ";R
60 IFR=0THEN130
```

```
65 IFR<0ORR>NRTHENPRINTCHR$(145);:GOTO55
70 REM PREPARA PUNTATORE
75 HI=INT(R/256):LO=R-HI*256
80 C$="P"+CHR$(LF)
85 C$=C$+CHR$(LO)+CHR$(HI)
90 C$=C$+CHR$(BI)
95 REM PREDISPONE PUNTATORE
100 PRINT#15,C$:GOSUB135
105 REM LEGGE RECORD
110 INPUT#LF,N$,T$:GOSUB135
115 PRINT"NOME:  ";N$
120 PRINT"TEL.:  ";T$
125 GOTO50
130 CLOSELF:GOSUB135:CLOSE15:END
135 REM ROUTINE ERRORE, AMMETTE COD. 50
140 INPUT#15,EN,EM$,ET,ES
145 IFEN=0OREN=50THENRETURN
150 PRINTEN;EM$;ET;ES
155 STOP:RETURN
```

Esso, dopo aver preparato le costanti e aperto il file, chiede il numero del record desiderato, calcola il puntatore, lo posiziona e va a leggere. Se vuoi leggere il file, come sequenziale, lo puoi fare con un solo posizionamento iniziale.

Segue il programma LEGGECAMPOREL, che invece di leggere un record, va a leggere il secondo campo di un record, dando un valore opportuno al puntatore.

```
5 REM LEGGICAMPOREL
10 REM LETTURA SOLO DEL SECONDO CAMPO
15 REM DI UN RECORD R
20 DR$="0":NF$=DR$+":PROVA REL,L,"
25 LU=21:SA=2:LF=2
30 BI=12:NR=30
35 INPUT"RECORD ";R
40 IFR=0ORR>NRTHENPRINTCHR$(145);:GOTO30
45 OPEN15,8,15,"I"
50 REM APREFILE
55 OPENLF,8,SA,NF$+CHR$(LU)
```

```
60 GOSUB120
65 REM PREPARA PUNTATORE
70 HI=INT(R/256):LO=R-HI*256
75 C$="P"+CHR$(LF)
80 C$=C$+CHR$(LO)+CHR$(HI)
85 C$=C$+CHR$(BI)
90 REM POSIZIONA PUNTATORE
95 PRINT#15,C$:GOSUB120
100 REM LEGGE CAMPO
105 INPUT#LF,T$:GOSUB120
110 PRINT"TEL.: ";T$
115 CLOSELF:GOSUB120:CLOSE15:END
120 REM ROUTINE ERRORE, AMMETTE COD. 50
125 INPUT#15,EN,EM$,ET,ES
130 IFEN=0OREN=50THENRETURN
135 PRINTEN;EM$;ET;ES
140 STOP:RETURN
```

Vogliamo farti riflettere sul fatto che il sistema, per consentirti di scrivere, per esempio, il quinto record del file, deve:

- determinare a quale settore appartiene il record,
- leggere il settore in un buffer,
- predisporre il puntatore nella giusta posizione,
- trasferire nel buffer i dati che tu scrivi,
- riscrivere sul dischetto l'intero settore.

Tutte queste operazioni per te risultano trasparenti, ma vengono eseguite. Nel caso dei file random, invece, queste operazioni le deve predisporre il programmatore nel suo programma.

La comodità di uso dei file relativi si paga con una maggiore occupazione di spazio disco, al massimo i sei settori dedicati ai side sector.

Anche per i file relativi sussiste il problema del reperimento del record in base ad un campo chiave ed è necessario ricorrere ad un file indice sequenziale, come per i file random. Solo che, in questo caso, nell'indice deve comparire la chiave del record, e, come indirizzo, il suo numero d'ordine nel file.

3.13 Esempio di archivio RELATIVO

Abbiamo modificato il programma ARCHIRANDOM per costruire il programma ARCHIRELATIVO.
Anche in questo caso abbiamo creato, in fase di generazione del file, un file indice sequenziale INDI1, avente come chiave di ordinamento i primi due campi del record logico: cognome e nome. L'indirizzo del record è il suo numero d'ordine nel file, in conseguenza il record del file indice INDI1 è formato solo da due campi, chiave e numero d'ordine.
È possibile costruire un indice secondario INDI2, con le stesse caratteristiche di quello dei programmi esempio precedenti.

Nella fase di creazione dell'archivio ex-novo, viene preesteso il file relativo al numero dei record richiesto, e viene creato il file indice INDI1. Il file indice INDI1 deve sempre essere riscritto tutto sul dischetto, per non perdere indirizzi. Questa volta abbiamo modificato il modo di gestire i dati generali dell'archivio; abbiamo creato un piccolo file sequenziale, di nome DATIGEN, che contiene il nome del file relativo, la data di aggiornamento, il numero totale di record previsti e il numero di record in essere. Il file DATIGEN viene letto all'inizio dell'elaborazione e riscritto alla fine.

```
5 REM ARCHIRELATIVO
10 REM---------------------------------
15 REM DEFINIZIONE COSTANTI E VARIABILI
20 REM---------------------------------
25 NC=14:REM **NUMERO CAMPI RECORD**
30 REM      ***********************
35 S1$="DATA ULTIMO AGG.      "
40 S2$="FINITO SPAZIO ASSEGNATO"
45 SP$="                             "
46 SWW=0:REM PER EVITARE RIDIMENSIONAMENTO
50 CH$=CHR$(13):LIM$=CHR$(99)+CHR$(99)+CHR$(99)
55 DIMD$(NC):DIMY$(NC):DIML(NC)
60 DATA"COGNOME:","NOME    :","INDIR. :"
65 DATA"CITTA' :","PROV.   :","TELEF. :"
70 DATA"L.NASC.:","D.NASC.:","T.STUD.:"
75 DATA"OCC.AT.:","OCC.PR.:","ST.CIV.:"
80 DATA"NOTA 1 :","NOTA 2  "
85 DATA20,15,30,25,11,10,20,8,20,20,20,1,20,20
90 FORJ=1TONC:READD$(J):NEXTJ
95 FORJ=1TONC:READL(J):NEXTJ
```

```
100 REM **PRESENTAZIONE MENU' E SCELTE**
105 REM ********************************
110 CLOSE15:OPEN15,8,15
115 PRINT"▮▮▮";TAB(10);"GESTIONE ARCHIVIO":PRINT
120 PRINTTAB(10)"1=INIZIO EX-NOVO"
125 PRINTTAB(10)"2=AGGIORNAMENTO"
130 PRINTTAB(10)"3=LISTA PRINCIPALE"
135 PRINTTAB(10)"4=CREAZ.IND.SEC."
140 PRINTTAB(10)"5=LISTA SECONDARIA"
145 PRINTTAB(10)"9=FINE"
150 INPUT"COSA SCEGLI ";X
155 IFX<1ORX>5ANDX<>9THENPRINT"]";:GOTO150
160 PRINT:GOSUB675
165 R$="":PRINTTAB(10)"MONTA DISCO DATI"
170 GETR$:IFR$=""THEN170
175 IFX=1THEN210
180 PRINT#15,"I"
185 GOSUB580:PRINT"PRESENTI ";K;" RECORD"
186 NN$=N$+",L,"+CHR$(254)
190 PRINTS1$;G1$;"/";M1$;"/";A1$
191 R$="":GETR$:IFR$=""THEN191
192 IFSWW=0THENDIMC$(N),T%(N)
193 GOSUB615
195 IFX=9THEN1340
200 ONX-1GOTO825,1150,1235,1290
205 STOP
210 REM-----------------------------
215 REM INIZIO EX-NOVO ARCHIVIO
220 REM-----------------------------
225 PRINT"POSSO FORMATTARE IL DISCO DATI S/N"
230 R$="":INPUTR$:IFR$=""THEN230
235 IFR$="S"THEN240
237 STOP
240 GOSUB740
250 INPUT"▮NOME FILE DA INIZIARE: ";N$
255 INPUT"QUANTI RECORD IN TUTTO: ";N
260 DIMC$(N),T%(N):SWW=1
261 REM PREPARAZIONE CAMPI DUMMY
263 P1$="*":FORJ=1TO52:P1$=P1$+" ":NEXTJ
265 P2$=LEFT$(P1$,37)
273 REM PREESTENSIONE FILE N$
274 OPEN11,8,11,N$+",L,"+CHR$(254)
```

```
275 FORJ=NTO1STEP-1
280 HI=INT(J/256):LO=J-HI*256
285 C$="P"+CHR$(11)+CHR$(LO)+CHR$(HI)+CHR$(1)
290 PRINT#15,C$:GOSUB645
293 PRINT#11,P1$CH$P1$CH$P1$CH$P1$CH$P2$CH$;
294 GOSUB645
295 NEXTJ:CLOSE11:GOSUB645
300 REM PREPARAZIONE INDICE
305 FORJ=1TON:T%(J)=J
310 C$(J)=LIM$:NEXTJ
315 GOSUB390:REM SCRIVE INDI1
320 K=0:GOSUB485:GOTO110
325 REM **INGRESSO DATI**
330 REM *****************
335 PRINT"□INGRESSO DATI"
340 FORJ=1TONC:Y$(J)="":NEXTJ
345 PRINT"PER USCIRE $ PER COGNOME";
350 PRINT"▮SE MANCANO DATIPREMI SOLO RETURN"
355 PRINTD$(1);:INPUTY$(1)
360 IFY$(1)="$"THENW=1:RETURN
365 FORJ=2TONC:PRINTD$(J);:INPUTY$(J):NEXTJ
370 REM **SISTEMA LUNGHEZZA DATI**
375 REM **************************
380 FORJ=1TONC:Y$(J)=LEFT$(Y$(J)+SP$,L(J))
385 NEXTJ:GOSUB1020:RETURN
390 REM **SCRITTURA INDICE PRINCIPALE**
395 REM *******************************
400 PRINT#15,"S0:INDI1":PRINT#15,"I"
405 OPEN10,8,10,"INDI1,S,W":GOSUB645
410 FORJ=1TON
415 PRINT#10,C$(J);CH$;T%(J);CH$;
420 GOSUB645:NEXTJ
425 CLOSE10:RETURN
430 REM **SCRITTURA RECORD PUNTATO**
435 REM ****************************
437 REM **COSTRUZIONE STRINGA DI STAMPA**
438 REM *********************************
440 C$="":FORJ=1TONC
443 C$=C$+Y$(J)+CH$:NEXTJ
445 PRINT#11,C$;:GOSUB645
450 RETURN
455 REM **DEFINISCE POSIZIONE RECORD K**
```

```
460 REM ******************************
465 HI=INT(T/256)
470 LO=T-HI*256
475 PRINT#15,"P"+CHR$(11)+CHR$(LO)+CHR$(HI)+CHR$(1)
480 GOSUB645:RETURN
485 REM **SCRITTURA DATIGEN DISCO**
490 REM **************************
495 PRINT#15,"S0:DATIGEN"
500 PRINT#15,"I"
505 OPEN10,8,10,"DATIGEN,S,W":GOSUB645
510 PRINT#10,N$CH$G$M$A$CH$;N;CH$;K;CH$;
515 CLOSE10:RETURN
520 REM **RICERCA POSTO NELL'INDICE**
525 REM ***************************
530 JJ=0:FORJ=1TON
535 IFLEFT$(C$(J),1)=CHR$(99)THEN 545
540 NEXTJ:RETURN
545 JJ=J:J=N:GOTO540
550 REM **LETTURA RECORD**
555 REM *****************
560 PRINT#15,"I":OPEN11,8,11,NN$:GOSUB645
565 GOSUB455
570 FORJ=1TONC:INPUT#11,Y$(J):GOSUB645
575 NEXTJ:CLOSE11:RETURN
580 REM **LETT.DATI DISCO**
585 REM ******************
590 OPEN10,8,10,"DATIGEN,S,R":GOSUB645
595 INPUT#10,N$,R$,N,K:GOSUB645
600 CLOSE10:GOSUB645
605 G1$=LEFT$(R$,2):M1$=MID$(R$,3,2)
610 A1$=RIGHT$(R$,2):RETURN
615 REM **LETT. INDICE**
620 REM ***************
625 OPEN10,8,10,"INDI1,S,R":GOSUB645
630 FORJ=1TON
635 INPUT#10,C$(J),T%(J):GOSUB645
640 NEXTJ:CLOSE10:RETURN
645 REM **ROUTINE ERRORE**
650 REM *****************
655 INPUT#15,EN,EM$,ET,ES
660 IFEN=0OREN=50THENRETURN
665 PRINT"█ERRORE DISCO"
```

```
670 PRINTEN,EM$,ET,ES:CLOSE15:STOP
675 REM **DATA DISCO**
680 REM **************
685 PRINT"DATA PER DISCO"
690 INPUT"  GG,MM,AA██████████";G$,M$,A$:RETURN
695 REM **SCRITT. IND. SEC.**
700 REM **********************
705 PRINT#15,"S:INDI2":PRINT#15,"I"
710 OPEN10,8,10,"INDI2,S,W"
715 FORJ=1TOK
720 PRINT#10,C$(J);CH$;T%(J);CH$;
725 GOSUB645:NEXTJ:CLOSE10:RETURN
730 R$="":GETR$:IFR$=""THEN730
735 RETURN
740 REM---------------------------
745 REM INIZIALIZZAZIONE DISCO
750 REM---------------------------
755 PRINT"NOME DISCO";:INPUTN$
760 PRINT#15,"N0:"+N$+",99"
765 CLOSE15:OPEN15,8,15:PRINT#15,"I"
770 RETURN
775 REM **NUOVI DATI**
776 REM **************
777 OPEN11,8,11,NN$:GOSUB645
779 GOSUB325
780 IFW=1THENK=K-1:CLOSE11:GOTO1340
785 GOSUB520:IFJJ=0THENSTOP:REM STOP IMPOSS.
790 T=T%(JJ):GOSUB455:GOSUB430
795 REM **AGGIORNAMENTO INDICE**
800 REM *************************
805 C$(JJ)=Y$(1)+Y$(2)
810 K=K+1
815 IFK>NTHENK=K-1:PRINTS2$:CLOSE11:GOTO1340
820 GOTO779
825 REM----------------
830 REM AGGIORNAMENTO
835 REM----------------
840 PRINTTAB(10);"███AGGIORNAMENTO ARCHIVIO"
845 PRINT:PRINTTAB(10);"1=CORREZIONE"
850 PRINTTAB(10);"2=AGGIUNTA ELEM."
855 PRINTTAB(10);"3=CANCELL.ELEM."
```

```
860 PRINTTAB(10);"9=FINE"
865 INPUT"COSA SCEGLI ";X:IFX=9THEN1340
870 IFX<1ORX>3THEN840
875 IFX=2THEN1070
880 IFX=3THEN1110
885 GOTO970
890 REM **RICERCA RECORD**
895 REM ******************
900 PRINT"⊐";D$(1);:INPUTY$(1):W=0
905 IFY$(1)="$"THENW=1:RETURN
910 PRINTD$(2);:INPUTY$(2)
915 Y$(1)=LEFT$(Y$(1)+SP$,L(1))
920 Y$(2)=LEFT$(Y$(2)+SP$,L(2))
925 INPUT"QUALE OCCORRENZA ";X
930 IFX<=0THENPRINT"⌉";:GOTO925
935 FORJ=1TOK:IFC$(J)=Y$(1)+Y$(2)THEN955
940 NEXTJ
945 J=J-1:PRINT"NON TROVATO ";Y$(1);"  ";Y$(2)
950 GOSUB730:GOTO890
955 IFX<>1THENX=X-1:GOTO940
960 T=T%(J)
965 I=J:J=K:NEXTJ:RETURN
970 REM **CORREZIONE**
975 REM **************
980 GOSUB890:IFW=1THEN840
985 GOSUB550
990 REM **VA A MODIFICA DATI**
995 REM **********************
1000 GOSUB1020
1005 PRINT#15,"I":OPEN11,8,11,NN$
1006 GOSUB455:GOSUB430
1010 C$(I)=Y$(1)+Y$(2)
1015 CLOSE11:GOTO970
1020 REM **CONTROLLO DATI E MODIFICHE**
1025 REM ******************************
1030 PRINT"⊐CONTROLLO DATI E MODIFICHE"
1035 FORJ=1TONC:PRINTJ;" ";D$(J);" ";Y$(J):NEXTJ
1040 INPUT"▮TUTTO BENE S/N ";X$:IFX$="S"THENRETURN
1045 PRINT"QUALE CAMPO (0 USCITA)";:INPUTX
1050 IFX=0THENRETURN
1055 IFX>NCORX<1THENPRINT"⌉⌉":GOTO1040
1060 INPUTY$(X):Y$(X)=LEFT$(Y$(X)+SP$,L(X))
```

```
1065 GOTO1030
1070 REM **AGGIUNTA**
1075 REM ***********
1085 PRINT"]AGGIUNTA NUOVI RECORD"
1090 PRINT"PRESENTI ";K;" RECORD"
1095 PRINT"PUOI AGGIUNGERE ";N-K;" RECORD"
1100 GOSUB730
1105 W=0:K=K+1:GOTO775
1110 REM **CANCELLAZIONE**
1115 REM *****************
1120 M=0
1125 GOSUB890:IFW=1THEN1140
1130 X=L(1)+L(2):C$(I)=LEFT$(LIM$+SP$+SP$,X)
1135 M=M+1:GOTO1125
1140 REM SIST. INDICE
1145 GOSUB1385:K=K-M:GOTO1365
1150 REM------------
1155 REM LISTA FILE
1160 REM------------
1165 T$="LISTA PER INDICE PRIM."
1170 PRINT"ACCENDI STAMPANTE     PREMI UN TASTO"
1175 GETR$:IFR$=""THEN1175
1180 PRINT"]";T$:OPEN4,4
1185 PRINT#4:PRINT#4:PRINT#4,T$
1190 FORJ=1TO10:PRINT#4:NEXTJ
1195 L=2:FORI=1TOK:T=T%(I):GOSUB550
1200 PRINT#4,Y$(1);"  ";Y$(2)
1205 PRINT#4,Y$(3);"  ";Y$(4);"  ";Y$(5)
1210 FORM=6TONC:PRINT#4,D$(M);Y$(M):NEXTM
1215 PRINT#4:PRINT#4
1220 IFL=5THENPRINT#4:L=0
1225 L=L+1:NEXTI
1230 CLOSE4:GOTO1370
1235 REM---------------------------
1240 REM CREAZIONE INDICE SECONDARIO
1245 REM---------------------------
1250 PRINT"]INDICE SECONDARIO"
1255 INPUT"QUALE CAMPO ";X$
1260 X=VAL(X$):IFX<2ORX>NCTHENGOTO1255
1265 FORI=1TOK:T=T%(I):GOSUB550
1270 C$(I)=Y$(X):NEXTI
1275 GOSUB695:GOSUB615
```

```
1280 PRINT"FINITO IND. SEC."
1285 GOTO1370
1290 REM-------------------------------
1295 REM LISTA PER INDICE SECONDARIO
1300 REM-------------------------------
1305 PRINT"{CLR}LISTA IND. SEC."
1310 OPEN10,8,10,"INDI2,S,R":GOSUB645
1315 FORJ=1TOK:INPUT#10,C$(J),T%(J)
1320 GOSUB645:NEXTJ
1325 CLOSE10:GOSUB1450:GOSUB695
1330 T$="LISTA IND. SEC. "
1335 GOTO1170
1340 REM **CHIUSURA**
1345 REM ************
1350 PRINT"CHIUSURA ARCHIVIO"
1355 PRINT"{CLR}{DOWN} ORDINA INDICE INDI1"
1356 PRINT" ATTENDI CON PAZIENZA"
1360 GOSUB1385
1365 GOSUB390:GOSUB485
1370 PRINT"FINITO AGGIORNAMENTO"
1375 PRINT"SONO PRESENTI ";K;" RECORD"
1380 CLOSE15:STOP
1385 REM **ORDINAMENTO CRESCENTE**
1390 REM *************************
1395 L=N-1
1400 W=0
1405 FORJ=1TOL
1410 IFC$(J)<=C$(J+1)THEN1430
1415 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
1420 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1425 W=1
1430 NEXTJ
1435 IFW=0THENRETURN
1440 IFL=1THENRETURN
1445 L=L-1:GOTO1400
1450 REM **ORDINAMENTO DECRESCENTE**
1455 REM ***************************
1460 L=K-1
1465 W=0
1470 FORJ=1TOL
1475 IFC$(J)>=C$(J+1)THEN1495
1480 R$=C$(J):C$(J)=C$(J+1):C$(J+1)=R$
```

```
1485 X=T%(J):T%(J)=T%(J+1):T%(J+1)=X
1490 W=1
1495 NEXTJ
1500 IFW=0THENRETURN
1505 IFL=1THENRETURN
1510 L=L-1:GOTO1465
```

Non riteniamo di dover descrivere in dettaglio anche questo programma, ma ci limitiamo a segnalare i blocchi significativi per il tipo di file relativo.

SCRITTURA RECORD PUNTATO
linee 430-450
Viene eseguito dopo il blocco: DEFINISCE POSIZIONE RECORD (linee 455-480), che prepara il puntatore all'inizio di un determinato record. Come vedi prepariamo una stringa C$, che contiene la somma di tutte le stringhe e i caratteri separatori necessari per riempire il record. L'istruzione PRINT# 11,C$, scrive tutto il record logico.

INIZIO EX-NOVO ARCHIVIO
linee 210-320
Dopo aver formattato il dischetto, viene preesteso il file, generato il primo file indice INDI1 e generato il primo file DATIGEN.
Ti segnaliamo l'uso dello switch SWW per evitare il ridimensionamento dei vettori, linee 260 e 192.

RICERCA POSTO NELL'INDICE
linee 520-540
Serve per trovare il numero del record per un nuovo record da aggiungere.

Pensiamo che, per una completa comprensione delle operazioni possibili con i file su disco, sia molto utile studiare i quattro programmi di archivio presentati. Essi, ovviamente, potrebbero essere migliorati sotto diversi aspetti. Il più evidente è il possibile e facile miglioramento dei quadri video, facendo ricorso anche al colore; noi non ce ne siamo occupati, dato che il nostro argomento principale è di spiegare come si gestiscono i file. Lasciamo a te la cura e la soddisfazione di proseguire e migliorare il nostro lavoro. Vedi quanto detto in proposito nel primo volume.

Noi abbiamo provato il programa ARCHIRELATIVO generando il file ARCHIREL di 500 record logici. Riportiamo la directory del dischetto usato.

```
0 [illegible]
505   "ARCHIREL"              REL
20    "INDI1"                 SEQ
1     "DATIGEN"               SEQ
138 BLOCKS FREE.
```

Come puoi vedere, il file relativo occupa 505 settori, 500 per i record logici e 5 per i side sector necessari. Il file sequenziale DATIGEN occupa un solo settore. Il file INDI1 occupa solo 20 settori, infatti nella fase di impostazione le chiavi DUMMY sono solo di 3 caratteri; man mano che si aggiungono i record, le dimensioni di INDI1 crescono, ma non si arriverà mai ai 90 settori usati per l'esempio dell'archivio RANDOM, dato che in questo caso il record del file indice ha un campo meno.

3.14 Messaggi di errore del DOS

L'indicatore luminoso a luce rossa, posto sull'unità 1541, si accende quando è in corso una operazione di lettura o scrittura da o sul dischetto. Quando si verifica un errore, l'indicatore pulsa velocemente. Come abbiamo visto più volte, la routine di errore, richiedendo il contenuto di 4 variabili sul canale 15 può rilevare se si è verificato un errore, e, nello stesso tempo resetta (spegne) l'indicatore.
Le 4 variabili, che possono essere tutte di tipo stringa, ma, salvo la seconda, anche di tipo numerico, e che noi abitualmente chiamiamo EN,EM$,ET,ES, contengono:
EN, il numero dell'errore o del messaggio di avvertimento,
EM$, il messaggio, se esiste,
ET, la traccia coinvolta,
ES, il settore coinvolto.

Il messaggio di codice 0, significa che è andato tutto bene, mentre quello di codice 1, segnala il numero di file cancellati in una operazione di SCRATCH.

Riportiamo l'elenco degli altri messaggi.

20 READ ERROR
Il controller del disco non riesce a trovare la testata (header) del settore richiesto; o l'indirizzo passato non è valido, o il dischetto è rovinato.

21 READ ERROR
Il controller del disco non trova il carattere di sincronizzazione sulla traccia richiesta. Può essere allineata male la testina di lettura/scrittura, mancare, essere inserito male o non essere formattato il dischetto. Può trattarsi di un guasto hardware.

22 READ ERROR
È stata richiesta una operazione di tipo BLOCK, ma non può essere eseguita, o perchè il blocco è danneggiato o perchè l'indirizzo non è valido.

23 READ ERROR
Dopo la lettura di un blocco il controllo di CHECKSUM dà errore. Può dipendere da una difettosa messa a terra.

24 READ ERROR
Viene provocato un errore hardware a causa di una cattiva ricezione dei dati o di una testata. Può dipendere da una difettosa messa a terra.

25 WRITE ERROR
I dati registrati e i dati in memoria non coincidono.

26 WRITE PROTECT ON
Si tenta di scrivere su un dischetto protetto, con la finestra chiusa.

27 READ ERROR
Il controller del disco ha trovato un errore nella lettura di una testata e non ha letto il blocco. Può dipendere da una difettosa messa a terra.

28 WRITE ERROR
Il controllore cerca, dopo aver scritto un blocco, il carattere di sincronizzazione del blocco seguente e non lo trova entro il tempo prefissato. Il dischetto può essere rovinato o formattato male, o si è verificato un guasto hardware.

29 DISK ID MISMATCH
Indica il tentativo di accedere a un dischetto che non è stato inizializzato o che è rovinato.

30 SYNTAX ERROR
Il DOS non riesce a interpretare un comando ricevuto; probabilmente sono errati alcuni parametri o ne mancano.

31 SYNTAX ERROR
Il DOS non riconosce un comando; può dipendere da uno spazio prima del comando.

32 SYNTAX ERROR
Il comando inviato supera la massima lunghezza consentita.

33 SYNTAX ERROR
Il nome usato in OPEN o SAVE non è valido.

34 SYNTAX ERROR
Il DOS non riconosce il nome del file; per esempio, mancano i due punti dopo il numero dell'unità.

39 SYNTAX ERROR
Il comando inviato sul canale 15 non viene riconosciuto.

50 RECORD NOT PRESENT
Si tenta di leggere da un file dopo aver raggiunto la fine del file. Nel caso dei file relativi può essere ignorato se si sta estendendo il file.

51 OVERFLOW IN RECORD
Si tenta di scrivere in un blocco più caratteri di quelli possibili. Per esempio, non si è tenuto conto dei caratteri separatori tra i campi.

52 FILE TOO LARGE
Si scrivono record su un file relativo producendo overflow.

60 WRITE FILE OPEN
Si apre in lettura un file che è stato scritto e non chiuso.

61 FILE NOT OPEN
Si tenta un'operazione su un file che non è stato aperto. In alcuni casi può non comparire il messaggio, però l'operazione non viene eseguita.

62 FILE NOT FOUND
Il file richiesto non esiste.

63 FILE EXISTS
Si cerca di creare un file che esiste già e non si è usato il carattere "@" prima del nome.

64 FILE TYPE MISMATCH
Il tipo di operazione si riferisce a un file di tipo diverso da quello registrato.

65 NO BLOCK
Indica che il blocco richiesto con B-A è già occupato, ma fornisce in ET e ES il numero di traccia e settore del primo bblocco disponibile. Se ET e ES sono nulli,

significa che il dischetto è pieno.

66 ILLEGAL TRACK AND SECTOR
Il DOS tenta di accedere a settori che non esistono.

67 ILLEGAL SYSTEM T OR S
Indica una traccia e un settore illegali per il sistema.

70 NO CHANNEL
Indica che il canale richiesto è già occupato o che non ci sono canali liberi.

71 DIRECTORY ERROR
Indica che i controlli sul dischetto non permettono di creare una BAM valida. Probabilmente si deve perdere il contenuto e riformattare il dischetto.

72 DISK FULL
Indica o che il disco è pieno, o che non ci sono più entrate libere nella directory.

73 DOS MISMATCH (73, CBM DOS V2.6 1541) Le versioni 1 e 2 del DOS creano dischetti compatibili in lettura, ma non in scrittura. Si cerca di scrivere su un dischetto formattato con l'altra versione.

74 DRIVE NOT READY
Si tenta di lavorare sull'unità spenta.

Non tutti i messaggi di errore vengono segnalati spontaneamente dal sistema. Per questo ti raccomandiamo di usare sempre la routine di errore nei tuoi programmi.

3.15 Programmi di utilità

I programmi di utilità sono di aiuto durante il lavoro di programmazione. Nella scatola dell'unità 1541 è contenuto un dischetto TEST/DEMO che contiene alcuni programmi di utilità e alcuni programmi esempio. Riportiamo la directory del TEST/DEMO.

```
0  [illegible]
13    "HOW TO USE"           PRG
5     "HOW PART TWO"         PRG
4     "VIC-20 WEDGE"         PRG
1     "C-64 WEDGE"           PRG
4     "DOS 5.1"              PRG
```

```
11    "COPY/ALL"            PRG
9     "PRINTER TEST"        PRG
4     "DISK ADDR CHANGE"    PRG
4     "DIR"                 PRG
6     "VIEW BAM"            PRG
4     "CHECK DISK"          PRG
14    "DISPLAY T&S"         PRG
9     "PERFORMANCE TEST"    PRG
5     "SEQUENTIAL FILE"     PRG
13    "RANDOM FIAL"         PRG
558 BLOCKS FREE.
```

I primi due programmi: HOW TO USE e HOW PART TWO, mostrano sul video le spiegazioni relative agli altri programmi del dischetto. Puoi caricare il primo, dare RUN e seguire le istruzioni che compaiono sul video. Al termine del primo viene caricato automaticamente il secondo.

Il programma C-64 WEDGE serve per caricare il DOS 5.1, se non è già stato caricato prima. Dopo aver dato RUN, vedi comparire READY, e sono attivi i seguenti nuovi comandi:

/nome-programma, per caricare un programma da dischetto;

> oppure @, per mostrare lo stato delle 4 variabili di errore;

> $ oppure @$, per vedere la directory sul video, SENZA AZZERARE IL PROGRAMMA PRESENTE IN MEMORIA.
Il programma resta attivo fino a quando togli corrente al calcolatore.

Il programma COPY/ALL consente di fare la copia dei dischetti usando due unità 1541 collegate al calcolatore, una con numero 8 e l'altra con numero 9. Per poter cambiare a una delle unità il numero da 8 a 9, puoi usare il programma DISK ADDR CHANGE, seguendo le istruzioni che compaiono sul video. Ricorda che il cambio del numero dell'unità resta valido fino a quando togli corrente. Per ottenere una unità che risponda sempre al numero 9, si deve modificare via hardware il numero, operando all'interno.
Ti consigliamo, se usi COPY/ALL, o altro programma di copia (ne esistono in commercio, che lavorano con una sola unità collegata), di proteggere la finestra del disco sorgente per lavorare senza pericoli. Usando due unità per copiare dischetti, l'operazione risulta più veloce. Con una sola unità, si devono alternare i dischetti alcune volte, fino al compimento dell'operazione.

Il programma PRINTER TEST consente di provare le caratteristiche della stampante; ti conviene usarlo, almeno una volta, per renderti conto di quali possibilità di stampa disponi.

Il programma DIR è abbastanza interessante anche per le tecniche di programmazione usate. Puoi listarlo sulla stampante o sul video e seguire il listato leggendo il nostro commento.
Viene mostrato il MENU':

D-DIRECTORY
>-DISK COMMAND
Q-QUIT PROGRAM
S-DISK STATUS

per la scelta. Q fa uscire dal programma.
Il canale comandi viene aperto con lfn=2 all'inizio. Scegliendo D, il programma va alla linea 10 e apre la DIRECTORY come file "$0", sul canale 0, con lfn=1. Poi con GET#1, viene letta e stampata la directory carattere per carattere. Questo è un modo diverso dall'abituale per ottenere la stampa della directory.
Rispondendo S, viene letto, in modo diverso dal solito, lo stato delle 4 variabili d'errore. Vengono eseguite sul canale comandi, lfn=2, delle GET#2, fino ad incontrare RETURN, e si stampano i caratteri. Trovi questo alle linee 5000-5020.
Rispondendo >, il programma lo stampa come PROMPT e si mette in attesa di una stringa comando DOS; quando premi RETURN, viene eseguito il tuo comando con la PRINT#2 della linea 4020.

Il programma VIEW BAM mostra sul video la BAM del dischetto, dapprima le tracce da 1 a 17 e poi le tracce da 18 a 35. La BAM viene vista come una matrice, sull'asse orizzontale sono riportate le tracce e sull'asse verticale i settori. Le posizioni scure della matrice indicano i blocchi occupati. Il programma presenta una imperfezione, per le tracce da 18 a 24 sono mostrati, senza l'annullamento N/L, ma come occupati, anche i settori numero 19, che non sono disponibili.

Il programma CHECK DISK consente di controllare se un dischetto ha settori rovinati, dove non si riesce a scrivere. All'inizio il programma esegue la VALIDATE, quindi cancella i settori non registrati nella directory. Dopo il programma scrive nei settori liberi, usando le funzioni dirette del DOS, e alloca i settori dove scrive; tali settori non sono però registrati nella directory. Se non riesce a scrivere in un settore, ne annota l'indirizzo in una doppia tabella di 100 elementi. Alla fine segnala gli indirizzi dei settori guasti. In realtà il dischetto alla fine risulta tutto occupato nella BAM, settori buoni e non buoni, senza corrispondenza con la directory. Per poter usare il dischetto si deve rifare la VALIDATE e andare ad

occupare stabilmente, sia nella BAM che nella Directory, i settori guasti. Per occupare la BAM basta usare il comando B-A, per occcupare la directory si può andare a scriverci dentro direttamente, usando il comando B-P e U2. Il programma risulta molto lento, e questo dipende da come vengono ricercati i settori liberi su cui andare a scrivere.

Il programma DISPLAY T&S è molto utile per controllare cosa succede sul disco. Noi, come hai potuto vedere leggendo questo capitolo, lo usiamo molto. Esso consente di visualizzare, o sul video o sulla stampante, il contenuto di un settore carattere per carattere. La visualizzazione avviene in due modi, con i codici esadecimali dei caratteri e con i caratteri di stampa. La parte in codice esadecimale, operando la conversione decimale, fornisce i codici ASCII abituali, la parte in caratteri di stampa risulta poco chiara, dato che alcuni codici non danno luogo a caratteri stampabili. Il programma in certi casi va in crisi. Basta premere RUN-/STOP e RESTORE e poi RUN per ripartire.

Il programma PERFORMANCE TEST serve per provare il buon funzionamento del dischetto.

Gli ultimi due programmi: SEQUENTIAL FILE e RANDOM FIAL, sono due esempi di creazione e gestione di file sequenziale e random.

Noi abbiamo preparato qualche altro programma di utilità che riportiamo qui.

Il programma DCOMEFS, stampa la directory, leggendola come file sequenziale, carattere per carattere. La stampa viene predisposta per evidenziare i contenuti della directory. La prima parte della stampa mostra i 35 gruppi (uno per ogni traccia) di 4 byte dedicati alla BAM; il primo byte di ogni quartina segnala il totale dei blocchi liberi nella traccia, gli altri sono usati in sequenza per il primo gruppo di 8 settori, per il secondo gruppo di 8 settori e per i settori restanti. Nella seconda parte della stampa sono evidenziate le altre informazioni; per i file nell'ordine: tipo file, indirizzo primo blocco file, nome file, tre byte significativi per i file REL (indirizzo del primo blocco side sector e lunghezza record), sono saltati 4 byte non usati, traccia e settore sostitutivi per uso di OPEN@, numero dei blocchi del file in 2 byte.

```
5 REM DCOMEFS
10 T$(0)="DEL":T$(1)="SEQ":T$(2)="PRG"
15 T$(3)="USR":T$(4)="REL"
20 CLOSE15:OPEN15,8,15,"I"
25 OPEN2,8,2,"$":REM APERTURA FILE DIRECTORY
30 OPEN4,4:REM APERTURA STAMPANTE
35 PRINT#4,"STAMPA DIRECTORY COME FILE SEQUENZIALE"
40 PRINT#4
```

```
45 REM LETTURA PRIMI 2 BYTE
50 I=2:GOSUB230
55 REM LETTURA BAM
60 FORK=1 TO 35:PRINT#4,K;"    ";
65 I=4:GOSUB230
70 NEXTK:PRINT#4
75 REM LETTURA NOME DISCO
80 I=18:GOSUB205
85 PRINT#4,N$;"   ";
90 REM LETTURA ID
95 I=2:GOSUB205:PRINT#4,N$
100 REM LETTURA ALTRI 7 BYTE
105 I=7:GOSUB230
110 PRINT#4
115 REM LETTURA ALTRI 85 BYTE
120 FORK=1TO85:GET#2,A$:NEXTK
125 PRINT#4
130 FORJ=1TO8
135 GET#2,T$,A$,B$
140 IFSTTHENCLOSE2:CLOSE4:CLOSE15:STOP
145 IFT$=""THENT$=CHR$(128)
150 I=16
155 GOSUB205
160 PRINT#4,T$(ASC(T$)-128);"   ";
165 PRINT#4,ASC(A$+CHR$(0));ASC(B$+CHR$(0));
170 PRINT#4,N$
175 I=3:GOSUB230
180 FORK=1TO4:GET#2,A$:NEXTK
185 I=2:GOSUB230
190 I=2:GOSUB230
195 IFJ<8THENGET#2,A$,A$
200 NEXTJ:GOTO130
205 REM COSTRUZIONE STRINGA
210 N$="":FORL=1TOI
215 GET#2,L$
220 IFL$<>CHR$(96)THENIFL$<>CHR$(160)THENN$=N$+L$
225 NEXTL:RETURN
230 REM LETTURA E STAMPA GRUPPO BYTE
235 FORL=1TOI
240 GET#2,A$:PRINT#4,ASC(A$+CHR$(0));
245 NEXTL:PRINT#4
250 RETURN
```

Non riportiamo i risultati che puoi ottenere tu. Se vuoi, puoi modificare la linea 30 per trasferire i risultati sul video, così: OPEN4,3. Seguendo le indicazioni sulla struttura della directory, puoi costruire un programma che ti permetta anche di modificarla, andando a scriverci dentro in modo diretto, con i comandi del DOS.

Segue il programma CONTRDISCO; esso controlla che i file di tipo SEQ e di tipo PRG siano registrati bene andando a confrontare la somma dei blocchi concatenati con il numero totale dei blocchi riportato nella directory. Salta nel controllo i file di altro tipo. Ogni volta che controlla un blocco, fa apparire un pallino sul video. Qualora per un file controllato vengano riscontrate irregolarità, il programma chiede se si vuole cancellarlo. Se la risposta è S, il file viene cancellato e viene eseguita la VALIDATE del dischetto; se la risposta è N il programma si ferma, dopo aver chiuso tutti i file.

```
5 REM CONTRDISCO
10 REM --COSTANTI E VARIABILI--
15 M1$="**************":FI=0
20 M2$="█FILE NON CORRETTO: █"
25 M3$="█TUTTO IN ORDINE   ...█ SPERO !"
30 REM --APRE CANALE 15 CON LFN=1--
35 OPEN1,8,15,"I"
40 REM --APRE CANALE 2 PER DIRECTORY--
45 OPEN2,8,2,"#"
50 REM --APRE CANALE 3 PER FILE--
55 OPEN3,8,3,"#"
60 REM --FUNZ. PER PUNTARE ENTRATE DIRECTORY--
65 DEFFNF(X)=2+32*X
70 REM --PRIMO BLOCCO DIRECTORY--
75 TD=18:SD=1:GOSUB350
80 REM --ELABORA UNA ENTRATA--
85 FI$="":CB=0
90 REM --PUNTATORE A INIZIO ENTRATA--
95 PRINT#1,"B-P:2,"FNF(FI)
100 REM --LEGGE TIPO FILE CHE VA IN TY--
105 GET#2,A$:GOSUB385:TY=ASC(A$)
110 REM --LEGGE IND. INIZIO FILE, TR E SC--
115 GET#2,A$,B$:GOSUB380:TR=ASC(A$):SC=ASC(B$)
120 REM --LEGGE NOME FILE--
125 FORI=1TO16:GET#2,A$:FI$=FI$+A$
```

```
130 IFA$=CHR$(160)THENFI$=LEFT$(FI$,I-1):I=16
135 NEXT
140 REM --PUNTATORE AI 2 BYTE CONT. BLOCCHI--
145 PRINT#1,"B-P:2,"FNF(FI)+28:GET#2,A$,B$
150 REM --NB=VCONTATORE BLOCCHI--
155 GOSUB380:NB=ASC(A$)+ASC(B$)*256
160 REM --CONTROLLO TIPO FILE 1 O 2--
165 IFTY<>0THENTY=TY-128
170 IFTY<>2ANDTY<>1THEN310
175 IFTY<>2ANDTY<>1THEN85
180 REM --CONTROLLO BLOCCHI FILE--
185 PRINT"]QQQ"M1$
190 PRINT"CONTROLLO FILE":PRINTM1$"QQQ"
195 PRINTTAB(10)FI$"QQ"
200 REM --LEGGE BLOCCO FILE--
205 PRINT#1,"B-R:3,0,"TR,SC
210 REM --LEGGE BYTE CONCATENAMENTO--
215 PRINT#1,"B-P:3,0"
220 CB=CB+1:GET#3,A$,B$:GOSUB380
225 REM --STAMPA UN PALLINO PER OGNI BLOCCO--
230 TR=ASC(A$):SC=ASC(B$):PRINT"●";
235 IFTR<>0THEN205
240 REM --CONTROLLO LUNGHEZZA FILE--
245 IFCB=NBTHEN325
250 REM --CICLO DI ATTESA--
255 IFCB=NBTHENFORI=1TO250:NEXTI:GOTO85
260 PRINT:PRINTM2$FI$
265 REM --SE CONTEGGIO ERRATO--
270 PRINT"QDEVO AZZERARE  ? (RS■/RN■)"
275 GETA$:IFA$=""THEN275
280 IFA$="N"THEN400
285 IFA$<>"S"THEN275
290 PRINT"QAZZERAMENTO E VALIDATE"
295 PRINT#1,"S:"FI$:PRINT#1,"V"
300 CLOSE3:CLOSE2:CLOSE1:RUN
305 REM --PASSA A PROSSIMA ENTRATA--
310 FI=FI+1:IFFI=8THENGOSUB350:FI=0
315 GOTO175
320 REM --TUTTO BENE--
325 PRINT:PRINTTAB(9)"QQQ]OK":FI=FI+1
330 IFFI=8THENGOSUB350:FI=0
335 GOTO255
```

```
340 REM --RE LETTURA DIRECTORY--
345 REM --TD=0 A FINE DIRECTORY--
350 IFTD=0THENPRINTM3$:GOTO400
355 REM --LETTURA BLOCCO--
360 A$="":B$="":PRINT#1,"U1:2,0";TD;SD
365 GET#2,A$,B$:GOSUB380
370 TD=ASC(A$):SD=ASC(B$):RETURN
375 REM --SISTEMAZIONE BYTE LETTI--
380 IFB$=""THENB$=CHR$(0)
385 IFA$=""THENA$=CHR$(0)
390 RETURN
395 REM --CHIUSURA FILE--
400 CLOSE3:CLOSE2:CLOSE1:STOP
```

Il programma legge la directory e i file in modo diretto, usando gli indirizzi di traccia e settore. Abbiamo abbondato in REM, quindi non aggiungiamo commento. Ti suggeriamo di estendere il programma per fargli controllare anche i file di altro tipo, tenendo conto che:

- per poter controllare i file RANDOM, è necessario che essi siano registrati come USR nel modo da noi suggerito nel Paragrafo 3.11;
- per i file RELATIVI il controllo è semplice per quanto riguarda i blocchi del file principale, ma si deve andare a prelevare dall'entrata della directory anche l'indirizzo del primo blocco side sector e procedere con il controllo anche per questa parte.

Il programma TRAC/SET , che segue, è ricavato dal DISPLAY T&S del TEST-/DEMO, apportando alcune modifiche. Con esso si possono stampare gruppi di settori concatenati sui primi due byte o settori singoli.

```
5 REM TRAC/SET
10 RC$="        ■/■IDEO ■■■■■■■■■■D   ■■■P■RINTER"
15 RD$="        ■■VIDEO ■"
20 RE$="        ■■PRINTER■"
25 PRINT"■■■―――――――――――――――――――――――"
30 PRINT"  CONTENUTO BLOCCHI    "
35 PRINT"――――――――――――――――――――――"
40 REM COSTANTI
45 SP$=" ":NL$=CHR$(0)
50 HX$="0123456789ABCDEF"
```

```
55 FS$="":FORI=64 TO 95
60 FS$=FS$+"▟"+CHR$(I)+"▀":NEXT I
65 SS$="  ":FOR I=192 TO 223
70 SS$=SS$+"▟"+CHR$(I)+"▀":NEXT I
75 DIM A$(15),NB(2)
80 D$="0"
85 PRINTRC$
90 GETJJ$:IF JJ$="" THEN90
95 IF JJ$="V"THENPRINTRD$
100 IF JJ$="P"THENPRINTRE$:OPEN4,4
110 GOSUB450
115 REM CARICA BUFFER
120 INPUT"███TRACCIA, SETTORE";T,S
125 IF T=0 OR T>35 THEN355
130 IF JJ$="V" THENGOSUB360:GOTO135
131 PRINT#4:PRINT#4,"TRACCIA"T" SETTORE"S:PRINT#4
135 PRINT#15,"U1:2,"D$;T;S:GOSUB335
140 PRINT#15,"B-P:2,0"
145 REM LEGGE BYTE 0
150 GET#2,A$(0):IFA$(0)=""THENA$(0)=NL$
155 PRINT#15,"B-P:2,1"
160 IF JJ$="V"THEN170
165 IF JJ$="P"THEN230
170 REM VIDEO
175 K=1:NB(1)=ASC(A$(0))
180 FORJ=0TO63:IFJ=32THENGOSUB375:GOTO365
185 FOR I=K TO 3
190 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
195 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
200 NEXT I:K=0
205 A$="":B$=":":N=J*4:GOSUB 405:A$=A$+":"
210 FOR I=0 TO 3:N=ASC(A$(I)):GOSUB 405
215 C$=A$(I):GOSUB 425:B$=B$+C$
220 NEXT I:IF JJ$="V" THEN PRINTA$B$
225 NEXT J:GOTO295
230 REM PRINTER
235 K=1:NB(1)=ASC(A$(0))
240 FOR J=0 TO 15
245 FOR I=K TO 15
250 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
255 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
260 NEXT I:K=0
```

```
265 A$="":B$=":":N=J*16:GOSUB 405:A$=A$+":"
270 FOR I=0 TO 15:N=ASC(A$(I)):GOSUB 405
275 C$=A$(I):GOSUB 425:B$=B$+C$
280 NEXT I
285 IF JJ$="P" THEN PRINT#4,A$B$
290 NEXT J:GOTO295
295 REM SUCCESSIVO BLOCCO
300 PRINT"TRACCIA/SETT.SEGUENTE"NB(1)NB(2) "█"
305 PRINT"DESIDERI PROSEGUIRE          S/N    "
310 GET Z$:IF Z$="" THEN310
315 IF Z$<>"S" THEN325
320 T=NB(1):S=NB(2):GOSUB445:GOSUB450:GOTO125
325 IF Z$="N" THEN GOSUB445:GOSUB450:GOTO120
330 GOTO 310
335 REM ROUTINE ERRORE
340 INPUT#15,EN,EM$,ET,ES:IF EN=0 THEN RETURN
345 PRINT"█ERRORE DISCO█"EN,EM$,ET,ES
350 END
355 PRINT#15,"I"D$:GOSUB445:CLOSE4:PRINT"END":END
360 PRINT"███TRACCIA"T" SETTORE"S"█":RETURN
365 IF Z$="N"THEN J=80:GOTO 225
370 GOTO185
375 REM MESS. CONTINUAZIONE
380 PRINT"█████CONTINUO(S/N)"
385 GETZ$:IF Z$="" THEN 385
390 IF Z$="N" THEN RETURN
395 IF Z$<>"S" THEN 385
400 PRINT"█TRACCIA" T " SETTORE"S:RETURN
405 REM CONV.HEX
410 A1=INT(N/16):A$=A$+MID$(HX$,A1+1,1)
415 A2=INT(N-16*A1):A$=A$+MID$(HX$,A2+1,1)
420 A$=A$+SP$:RETURN
425 REMCONV.ASCII
430 IF ASC(C$)<32 THEN C$=" ":RETURN
435 IF ASC(C$)<128 OR ASC(C$)>159 THEN RETURN
440 C$=MID$(SS$,3*(ASC(C$)-127),3):RETURN
445 CLOSE2:CLOSE15:RETURN
450 OPEN15,8,15,"I"+D$:GOSUB335
455 OPEN2,8,2,"#":GOSUB335:RETURN
```

Il programma SIST/DISCO è una modifica sostanziale di CHECK DISK; esso lascia il dischetto in grado di essere usato e con allocati, solo nella BAM, i settori guasti. Su un dischetto in queste condizioni non può essere eseguita la funzione VALIDATE.

```
5 REM SIST/DISCO
10 REM TABELLA TRACCE E SETTORI GUASTI
15 DIMT(100):DIMS(100)
20 DIMG1(21),G2(19),G3(18),G4(17)
25 DATA0,10,20,8,18,6,16,4,14,2,12,9,19
30 DATA7,17,5,15,3,13,1,11
35 DATA0,11,1,12,2,13,3,14,4,15,5,16,6,17
40 DATA7,18,8,9,10
45 DATA0,10,1,11,2,12,3,13,4,14,5,15
50 DATA6,16,7,17,8,9
55 DATA0,10,1,11,2,12,3,13,4,14,5
60 DATA15,6,16,7,8,9
65 FORJ=1TO21:READG1(J):NEXTJ
70 FORJ=1TO19:READG2(J):NEXTJ
75 FORJ=1TO18:READG3(J):NEXTJ
80 FORJ=1TO17:READG4(J):NEXTJ
85 RC$=" BLOCCHI GUASTI SONO STATI ALLOCATI"
90 RD$="██BLOCCHI GUASTI":RE$="TRACCIA"
95 RF$="SETTORE"
100 PRINT"███────────────────────────"
105 PRINT"      SIST/DISCO      "
110 PRINT"────────────────────────"
115 OPEN15,8,15
120 PRINT#15,"V0"
125 N%=RND(TI)*255
130 A$="":FORI=1TO255
135 A$=A$+CHR$(255AND(I+N%)):NEXT
140 OPEN2,8,2,"#":GOSUB325
145 PRINT:PRINT#2,A$;
150 J=1
155 FORT=1TO17:FORL=1TO21
160 S=G1(L):GOSUB200:NEXTL:NEXTT
165 FORT=18TO24:FORL=1TO19
170 S=G2(L):GOSUB200:NEXTL:NEXTT
175 FORT=25TO30:FORL=1TO18
180 S=G3(L):GOSUB200:NEXTL:NEXTT
```

```
185 FORT=31TO35:FORL=1TO17
190 S=G4(L):GOSUB200:NEXTL:NEXTT
195 GOTO260
200 PRINT#15,"B-A:0"T;S
205 INPUT#15,EN,EM$,ET,ES
210 IFEN=0THEN225
215 IFEN=65THENRETURN
220 STOP
225 PRINT#15,"U2:2,0"T;S
230 NB=NB+1
235 INPUT#15,EN,EM$,ES,ET
240 IF EN=0THENRETURN
245 T(J)=T:S(J)=S:J=J+1
250 PRINT"▓▓▓BLOCCHI GUASTI:";T;S
255 RETURN
260 PRINT#15,"V0"
265 GOSUB325
270 CLOSE2
275 IFJ=1THEN350
280 OPEN2,8,2,"#"
285 PRINTRD$,RE$,RF$
295 FORI=1TOJ-1
300 PRINT#15,"B-A:0"T(I);S(I)
305 PRINT,,T(I),S(I)
310 NEXT
315 PRINT"▓";J-1;RC$
320 CLOSE2:CLOSE15:END
325 INPUT#15,EN,EM$,ET,ES
330 IF EN=0 THEN RETURN
335 PRINT"▓▓▓ERRORE#"EN,EM$;ET;ES"▓"
340 PRINT#15,"I0"
345 RETURN
350 PRINT"▓▓▓▓▓▓TUTTO BENE!     ":CLOSE15:END
```

La ragione che rende molto veloce questo programma sta nel modo ottimizzato di allocare e scrivere i diversi blocchi. Abbiamo preparato con le frasi DATA e i quattro vettori G1, G2, G3 e G4 una traccia ottimizzata per l'allocazione dei settori.

Segue il programma NUMBUFFER; esso serve per vedere quanti buffer possono essere aperti contemporaneamente in modo diretto e quali sono i loro numeri (da 0 a 7).

```
5 REM NUMBUFFER
10 OPEN15,8,15,"I"
15 INPUT"QUANTI BUFFER";N
20 A$="":B$="":C$="":D$="":E$=""
25 ONNGOTO50,45,40,35,30
30 OPEN2,8,2,"#":GOSUB115:GET#2,E$:GOSUB115
35 OPEN3,8,3,"#":GOSUB115:GET#3,D$:GOSUB115
40 OPEN4,8,4,"#":GOSUB115:GET#4,C$:GOSUB115
45 OPEN5,8,5,"#":GOSUB115:GET#5,B$:GOSUB115
50 OPEN6,8,6,"#":GOSUB115:GET#6,A$:GOSUB115
55 ONNGOTO100,90,80,70,60
60 IFE$=""THENE$=CHR$(0)
65 PRINTASC(E$):CLOSE2
70 IFD$=""THEND$=CHR$(0)
75 PRINTASC(D$):CLOSE3
80 IFC$=""THENC$=CHR$(0)
85 PRINTASC(C$):CLOSE4
90 IFB$=""THENB$=CHR$(0)
95 PRINTASC(B$):CLOSE5
100 IFA$=""THENA$=CHR$(0)
105 PRINTASC(A$):CLOSE6
110 CLOSE15:STOP
115 INPUT#15,EN,EM$,ET,ES
120 PRINTEN;EM$;ET;ES
125 RETURN
```

Come puoi vedere, provando il programma, se cerchi di aprire più di 4 canali, hai la segnalazione NO CHANNEL. L'istruzione GET#lfn, usata subito dopo la OPEN..."#", fornisce il numero del buffer assegnato. Se provi il programma dopo averlo caricato con LOAD trovi che, per 4 canali, vengono assegnati i buffer: 2, 1, 0, 4. Se, invece, dopo il LOAD del programma, togli corrente all'unità 1541, ottenendo il reset della RAM, poi ridai corrente e RUN, ottiene, sempre per 4 canali, i numeri di buffer 3, 2, 1, 0.

Segue il programma INDBUFFER; esso serve per vedere quali indirizzi della RAM dell'unità 1541 sono assegnati ai diversi buffer.

```
5 REM INDBUFFER
10 DIMR$(8,7):FORK=1TO8:FORI=1TO7
15 R$(K,I)="":NEXTI:NEXTK
16 M1$="IND. IN MEMORIA DEL BUFFER: "
20 OPEN7,4:PRINT#7,"RISULTATI INDBUFFER"
25 OPEN15,8,15,"I"
30 INPUT"QUANTI BUFFER";N:PRINT#7
35 PRINT#7,N;" BUFFER":PRINT#7
40 A$="":B$="":C$="":D$="":E$=""
45 ONNGOTO70,65,60,55,50
50 OPEN2,8,2,"#":GOSUB270:GET#2,E$:GOSUB270
55 OPEN3,8,3,"#":GOSUB270:GET#3,D$:GOSUB270
60 OPEN4,8,4,"#":GOSUB270:GET#4,C$:GOSUB270
65 OPEN5,8,5,"#":GOSUB270:GET#5,B$:GOSUB270
70 OPEN6,8,6,"#":GOSUB270:GET#6,A$:GOSUB270
75 PRINT#7,"NUMERI DEI BUFFER: ";
80 ONNGOTO165,145,125,105,85
85 IFE$=""THENE$=CHR$(0)
90 PRINT#7,ASC(E$);
95 PRINT#15,"B-P:"2;1
100 PRINT#2,"22222";:GOSUB270
105 IFD$=""THEND$=CHR$(0)
110 PRINT#7,ASC(D$);
115 PRINT#15,"B-P:"3;1
120 PRINT#3,"33333";:GOSUB270
125 IFC$=""THENC$=CHR$(0)
130 PRINT#7,ASC(C$);
135 PRINT#15,"B-P:"4;1
140 PRINT#4,"44444";:GOSUB270
145 IFB$=""THENB$=CHR$(0)
150 PRINT#7,ASC(B$);
155 PRINT#15,"B-P:"5;1
160 PRINT#5,"55555";:GOSUB270
165 IFA$=""THENA$=CHR$(0)
170 PRINT#7,ASC(A$);
175 PRINT#15,"B-P:"6;1
180 PRINT#6,"66666";:GOSUB270
185 PRINT#7
190 FORK=1TO8:M=K-1
195 FORI=1TO7:L=I-1
200 PRINT#15,"M-R"CHR$(L)CHR$(M)
205 GET#15,R$(K,I)
```

```
210 NEXTI
215 NEXTK
220 FORK=1TO8:FORI=1TO7
225 IFR$(K,I)=""THENR$(K,I)=CHR$(0)
230 NEXTI:NEXTK
235 FORK=1TO8
240 PRINT#7,M1$;STR$((K-1)*256)
245 PRINT#7,"PRIMI 8 BYTE DEL BUFFER:";
250 FORI=1TO7
255 PRINT#7,R$(K,I);:NEXTI:PRINT#7
260 NEXTK
265 FORK=1TO7:CLOSEK:NEXTK:CLOSE15:STOP
270 INPUT#15,EN,EM$,ET,ES
275 PRINTEN;EM$;ET;ES
280 RETURN
```

Il programma fa le seguenti cose:

- chiede quanti buffer si vogliono, apre il numero richiesto di file (massimo 5) e, per ognuno, legge con GET# il numero del buffer e lo memorizza;
- scrive per ogni file aperto 5 caratteri di riconoscimento nel buffer a partire dalla posizione 1;
- legge con M-R i primi 7 caratteri di ogni buffer (da 0 a 7), usando gli indirizzi iniziali: 0, 256, 512, 768, 1024, 1280, 1536, 1792 (i primi 4K di RAM) e li memorizza nella matrice R$(8,7);
- stampa i risultati.

Riportiamo i risultati ottenuti con due prove; la prima si riferisce a un RUN senza reset dell'unità 1541, la seconda a un RUN dopo il reset.

```
RISULTATI INDBUFFER

 4  BUFFER

NUMERI DEI BUFFER:  2  1  0  4
IND.. IN MEMORIA DEL BUFFER:  0
PRIMI 8 BYTE DEL BUFFER:
IND. IN MEMORIA DEL BUFFER:  256
PRIMI 8 BYTE DEL BUFFER:A
IND. IN MEMORIA DEL BUFFER:  512
PRIMI 8 BYTE DEL BUFFER:M-R
```

```
IND. IN MEMORIA DEL BUFFER:  768
PRIMI 8 BYTE DEL BUFFER:55555>
IND. IN MEMORIA DEL BUFFER:  1024
PRIMI 8 BYTE DEL BUFFER:44444N
IND. IN MEMORIA DEL BUFFER:  1280
PRIMI 8 BYTE DEL BUFFER:33333I
IND. IN MEMORIA DEL BUFFER:  1536
PRIMI 8 BYTE DEL BUFFER:
IND. IN MEMORIA DEL BUFFER:  1792
PRIMI 8 BYTE DEL BUFFER:
```

```
RISULTATI INDBUFFER

 4  BUFFER

NUMERI DEI BUFFER:  3  2  1  0
IND. IN MEMORIA DEL BUFFER:  0
PRIMI 8 BYTE DEL BUFFER:
IND. IN MEMORIA DEL BUFFER:  256
PRIMI 8 BYTE DEL BUFFER:A
IND. IN MEMORIA DEL BUFFER:  512
PRIMI 8 BYTE DEL BUFFER:M-R

IND. IN MEMORIA DEL BUFFER:  768
PRIMI 8 BYTE DEL BUFFER:66666
IND. IN MEMORIA DEL BUFFER:  1024
PRIMI 8 BYTE DEL BUFFER:55555
IND. IN MEMORIA DEL BUFFER:  1280
PRIMI 8 BYTE DEL BUFFER:44444
IND. IN MEMORIA DEL BUFFER:  1536
PRIMI 8 BYTE DEL BUFFER:33333
IND. IN MEMORIA DEL BUFFER:  1792
PRIMI 8 BYTE DEL BUFFER:
```

APPENDICE A

MAGIC DESK I

Questo package, realizzato in modo piuttosto divertente, consente di usare i file su disco anche senza saper scrivere programmi che li gestiscono.

Il package risiede su cartridge; dopo l'inserimento, a macchina spenta, quando dai corrente compare sul video un menù quadro. Vengono proposte le diverse funzioni sotto forma di disegni in piacevoli colori. Per usare il programma è necessario inserire un Joystick nella porta 2. I comandi si scelgono usando il Joystick per muovere nelle 4 direzioni la mano che compare sul video. Per scegliere una funzione si deve toccare con il dito della mano, mobile sul video, l'oggetto che la rappresenta; premendo poi il bottone del fuoco la funzione va in esecuzione.
Il sistema necessario per usare il package in tutte le sue possibilità è composto da: il calcolatore, l'unità disco, la stampante, il video e un joystick.

Per sapere cosa si può fare, basta premere il tasto CBM; compare sul video un menù in parte disegnato e in parte descritto. Devi inserire un dischetto nuovo, la prima volta che usi il programma, nell'unità 1541.

Le operazioni possibili sono:

- Scrivere a macchina usando la tastiera del calcolatore e vedendo sul videc quello che si scrive. Il calcolatore si trasforma in una vera macchina da scrivere, con attivabili tutte le normali funzioni di tale apparecchiatura.

- Predisporre l'orologio, che avanza in tempo reale, all'ora che si desidera.

- Memorizzare i documenti scritti (funzione di archiviazione) in uno dei 3 cassetti raccoglitori visibili, e in una determinata cartelletta.

• Richiamare sul video documenti già archiviati per correggerli, stamparli, e, se si vuole, conservarli di nuovo.

• Se un documento scritto non interessa più, si può buttarlo nel cestino della carta straccia.

Segue un listato di una lettera scritta e stampata con il programma.

Spett.
Lettore del libro
COMMODORE 64 - I FILE

Milano, giugno 1984

Caro Lettore,

spero che questo manuale ti sia utile, e che tu abbia imparato ad usare bene i file sia su cassetta che su disco.

Comunque, puoi sempre servirti di MAGIC DESK I per risolvere i tuoi problemi.

Gradisci i miei migliori saluti

Rita Bonelli

APPENDICE B

CALC RESULT

È un ottima versione del FOGLIO ELETTRONICO. Viene fornito un manuale esauriente, una ROM da inserire nel calcolatore e un dischetto.

L'utente ha a disposizione sul video una matrice formata da 64 colonne (denominate: a,b,c,...,aa,ab,ac,...,ba,bb,...,bk) e da 254 righe (denominate: 1,2,...,254). In realtà sul video è visibile solo una parte di questo grande foglio elettronico, ma, servendosi dei tasti di movimento del cursore, si può far comparire la parte che interessa della matrice.
In ogni casella della matrice si può scrivere:

- una LABEL (descrizione),
- un numero,
- una formula di calcolo.

Ogni casella può essere facilmente modificata; si può definire la larghezza in caratteri di ogni colonna della matrice.
Le formule di calcolo lavorano sulle altre caselle della matrice (che sono le variabili). Sono disponibili anche alcune funzioni definite dal sistema; tra esse MIN, MAX, IF...THEN...ELSE, funzioni matematiche e logiche.
Se si modifica un numero, contenuto in una casella coinvolta in una formula, automaticamente viene ricalcolato il risultato.
Il contenuto di un foglio elettronico può esssere salvato su disco e richiamato quando serve; esso può essere stampato tutto o in parte.
Sono gestibili contemporaneamente più fogli elettronici, fino a 32. È possibile costruire un foglio prendendo dati da altri fogli.
Il video può essere diviso in due parti (finestre) gestibili separatamente.

Riportiamo un semplicissimo esempio di bilancio familiare su 3 mesi.

	GENN.	FEBBR.	MARZO
AFFITTO	300000	300000	300000
VITTO	600000	650000	700000
LUCE	70000	80000	90000
GAS	20000	15000	18000
TELEF.	80000	90000	100000
AUTO	100000	120000	150000
SCUOLE	60000	60000	60000
TOTALE	1230000	1315000	1418000

Il programma si presta a svariate applicazioni come:

- bilanci,
- piani di ammortamento,
- ripartizione di spese,
- gestione del conto corrente,
- previsioni finanziarie,
- stampa di grafici.

Si possono creare dei modelli di matrice per diverse applicazioni, memorizzarli su disco con nomi opportuni e servirsene per lavorare.

Inoltre è possibile generare dei file di dati di tipo DIF (Data Interchange Format), che possono essere utilizzati da altri programmi scritti in BASIC.

APPENDICE C

LE BASI DI DATI

Sono disponibili sul mercato diversi package per la gestione sofisticata di archivi (basi di dati). Essi fanno le poche, ma fondamentali, cose, che noi abbiamo trattato nei capitoli precedenti, più tante altre. I programmi componenti il pacchetto sono scritti in ASSEMBLER, quindi la loro velocità di esecuzione non è paragonabile con quella dei nostri programmi in BASIC.

Vogliamo qui citare due package, che abbiamo avuto modo di provare; essi sono:

SUPERBASE 64 della Precision Software,
MAGPIE DATABASE, distribuito dalla JCE nella versione italiana.

Si tratta di due prodotti pregevoli; essi non solo consentono di generare e gestire archivi di dati con uno o più indici di ricerca, ma consentono anche di svolgere calcoli e di richiamare programmi che operano sui record degli archivi.

L'utente può predisporre i tracciati dei record dei file, i tracciati dei quadri video e i formati dei prospetti da stampare.

I manuali che accompagnano i programmi sono di notevole mole, e ci vuole un pò di tempo per impadronirsi di tutte le possibilità dei package. Si tratta comunque di prodotti che possono essere usati anche da persone inesperte di programmazione, ma non digiune dei concetti fondamentali inerenti il trattamento dei dati con il calcolatore.

Questo libro tratta in maniera completa e precisa la gestione dei file su cassetta e su disco sul COMMODORE 64.
Oltre a brevi programmi esempio, riportati per spiegare l'uso delle istruzioni, il libro contiene cinque programmi per creare e gestire un archivio di dati:

SEQUENZIALE su cassetta,
SEQUENZIALE su disco,
RANDOM su disco,
RANDOM/USER su disco,
RELATIVO su disco.

E' interessante poter fare dei confronti tra i cinque programmi; questo risulta di grande utilità per capire a fondo l'argomento trattato.

Prezzo L. 17.000

Cod. 400B ISBN 88-7056-056-2

GRUPPO
EDITORIALE
JACKSON

Rita Bonelli

COMMODORE 64 i file

124